The University of Illinois | College of Fine and Applied Arts | School of Music

MUS 499C: Introduction to Audio Coding with SuperCollider

Fall 2020 Course Syllabus

Basic Information

Course:	MUS 499C Advanced Studio Techniques II: Introduction to Audio Coding with SuperCollider
Credit:	2 undergraduate/graduate hours
CRN:	38243
Location and Time:	http://youtube.com/user/elifieldsteel, Thursdays 3-3:50pm CDT (UTC-5)
Instructor:	Dr. Eli Fieldsteel (he/him/his)
Course Website:	http://learn.illinois.edu

Course Description, Objectives, and Philosophy

This course introduces SuperCollider (SC), a free, open-source, cross-platform programming environment for real-time digital audio synthesis and algorithmic composition. SuperCollider is a member of a family of audio programming platforms that have been created and developed over the past few decades, which includes Csound, Max/MSP, Pure Data, Kyma, ChucK, RTcmix, Sonic Pi, TidalCycles, FoxDot, and many others. These platforms can be described as audio sandboxes; their objects, classes, and methods are specifically designed for sound applications, and exist within a highly flexible and modular environment. As a result, these types of programs offer considerably deeper potential for creative and experimental work compared to DAW software and other types of timeline/track-based software.

Why SuperCollider, then? In my opinion, SC offers a number of unique strengths among its siblings, all of which will hopefully become clear over the course of the semester. Besides being free, open-source, and cross-platform (Mac/Win/Linux), SC has a lean, powerful audio synthesis engine at its core, coupled with a client-server design and Open Sound Control (OSC) functionality, making it an exceptionally flexible and adaptable platform in many contexts. With practice and dedication, expressing musical intent through SC code becomes a fluid, rewarding, and liberating experience.

Well before COVID-19 stepped into the spotlight, I had decided to put the lecture component of this class online in a livestreaming format. One of the main advantages of this approach is that the burden of real-time note-taking is largely removed, as students can re-watch the archived livestream videos as often as needed to recall and internalize concepts. This is particularly useful for a course like this, which focuses on new and highly technical programming code. On the stream, students can comment and ask questions in real-time using the chat, and share code quickly using services like pastebin.com. While watching live or archived lecture video, students can simultaneously run SC on their own computers to follow along, if desired. In the past, I used face-to-face classroom time for programming exercises, giving students an opportunity to solve practical problems with direct access to the instructor for extra help. However, due to COVID-19, all homework assignments this semester will be administered online and completed remotely.

Generally speaking, computer programming and digital audio fluency are valuable skills across the contemporary job market. It is rarer and rarer to find jobs that require no programming skills whatsoever—or at least those in which an applicant with programming skills is considered less desirable than one without. While SC is distinct from more general languages like C++, Java, Python, etc., there are certain coding fundamentals that are shared among these platforms. Commercial and freelance digital multimedia projects frequently benefit from having a skilled programmer and/or creative sound designer on the team. This course is meant to improve abilities in computer programming and digital audio fluency, striving to help you become more nimble, adaptable, and competitive on the job market, and to help you understand these concepts on a deeper level.

After completing this course, it is my hope that you will not only become familiar with SC and its capabilities, but also develop an ability to identify and solve common audio programming problems, more fully understand the principles of common synthesis tools and algorithms, become familiar with designing simple graphical interfaces for sound, and generally become more deeply acquainted with avenues for coding electronic, algorithmic, and/or interactive sound.

Required Materials

- Access to a computer running macOS, Windows, or Linux, with the latest version of SuperCollider installed.
- Over-the-ear headphones or a pair of good quality loudspeakers. Earbuds are acceptable, but not recommended.
- A smartphone or tablet running iOS or Android (for downloading/using TouchOSC, $5)
- Access to a basic USB MIDI controller with some combination of keys, faders, buttons, and knobs. These are available for checkout through me, if needed.

SuperCollider Resources

While there is no textbook for this course, there are a variety of helpful learning resources online:

- Main page for SuperCollider: http://supercollider.github.io/ (includes download, wiki, tutorials, etc.)
- My ongoing SuperCollider Tutorial Series on YouTube: https://www.youtube.com/playlist?list=PLPYzvS8A_rTaNDweXe6PX4CXSGq4iEWYC — the video description for Tutorial 0 includes several links to resources
- The SC Forum: https://scsynth.org
- User-uploaded SC code examples: http://sccode.org/
- An archive of the SC-Users Mailing List, where you can search for answers to commonly asked questions: http://www.listarc.bham.ac.uk/marchives/sc-users/
- SuperCollider on Reddit: https://www.reddit.com/r/supercollider/
- The SuperCollider Book (MIT Press, 2011): https://mitpress.mit.edu/books/supercollider-book

It's difficult to overstate the importance of seeking help and asking questions when you find yourself struggling with course material, especially when dealing with novel complexities of digital audio programming. In addition to the resources listed above, please consider:

- asking questions in the chat during livestream lectures
- posting questions on the course Piazza page (see below), being sure to include as much relevant information as possible
- dropping into during office hours or emailing me to arrange a one-on-one meeting—I am a huge SC nerd and am always excited to help, especially if it allows me to procrastinate on other stuff
- if you're feeling really stuck, email me your code with as much information as possible, indicating how/where you are stuck, what error messages you are receiving, etc.

Piazza

Following a student suggestion from a previous semester, I've decided to use Piazza for class discussion. This is a free third-party platform that is highly catered to providing fast and efficient help from classmates and myself. If you have questions about the course, the assignments, or SC code in general, rather than emailing your questions to me, I encourage you to post your questions on Piazza. I'll be checking Piazza regularly and I encourage you to

do the same.

If you haven't already received an email invitation, you can find the class signup link here: https://piazza.com/illinois/fall2020/mus499c

Pastebin

Pastebin (http://pastebin.com) is a free website designed to make it easy to share large amounts of text—very handy for sending big chunks of code back and forth. There may be situations during a livestream in which my code is working but yours isn't. Rather than pasting your code directly into the YouTube live chat (which will likely obliterate your spacing, indentations, and other formatting), instead:

1. go to pastebin.com
2. paste your code into the "New Paste" field
3. click "Create New Paste"
4. copy the shortlink in your browser's URL bar and paste it in the YouTube chat

And I'll be able to put your code onstream within seconds.

Lecture

Lecture will be livestreamed to my YouTube channel: http://youtube.com/user/elifieldsteel. Lecture streams will be archived on my YouTube channel for future viewing. Note that the live chat is re-playable on archived stream videos. Streams and archives will be public.

Depending on a number of factors, I may need to move the lectures to Zoom or some other platform. In this unlikely event, lectures will continue to be recorded and uploaded for future viewing.

Homework Assignments

Programming problem sets will be assigned on a weekly basis. In the interest of fostering a communal learning experience, students are allowed to discuss the homework problems and collaborate outside of class. However, your work must ultimately be your own.

Including comments in your code is not required, but comments can be helpful in allowing me to better understand your intent. If you submit code that fails to work properly, I will probably have an easier time identifying the problem and explaining the issue if you provide comments.

In grading your homework, I will insert my own comments in your code and send it back to you. Each of my comments will be preceded with the string "EMF:" To facilitate the process of reviewing my comments, you can perform a text search for this string, and then easily step through each one.

Midterm Programming Exam

A take-home midterm programming exam will be assigned on Thursday October 8th at 4pm CDT. Collaboration with other students is prohibited (you are on the honor system). The exam will be due by October 9th at 4pm CDT via upload submission to the course website. Midterm exams received after the deadline will receive no credit, but there will be a 30-minute grace period in anticipation of slow internet connections, etc. Plan accordingly: start the exam early and don’t wait until the last minute to submit.

In cases of unexpected circumstances that prevent you from completing the exam as scheduled, I am willing to

work with you to arrange an alternative, in which case you must email me to explain your situation and discuss arrangements. I would like to avoid this, if possible.

Final Creative Project

Each student will propose and complete a creative project using SC as a central component, and present it to the class. You can do this live over Zoom, or make a presentation video ahead of time to be shared over Zoom. A group presentation date is TBD and will be announced in a timely manner. Your final creative project may be a composition, a live performance demonstration, a software tool for enhancing the compositional process, a software environment for improvisation, a game with a musical component, or something else. Final project proposals (a short written summary of your plan) will be due by 3pm CDT on November 5th.

Attendance and Participation

Due to COVID-19, attendance and participation will not be factored into your final grade, nor are you required to produce documentation to verify unexcused absences. I recognize the complexities and uncertainty of our global situation, and that there are many reasons why you may need to miss one or more days of class, request an extension, etc. In exchange for my flexibility and understanding, I ask that you make a good faith effort to engage yourself in the course to the greatest extent possible. I appreciate being notified of absences in advance via email, although this is not required.

If I note multiple absences, I may follow up with an email and/or submit an Irregular Attendance report. This is done out of care and concern for your well-being, and is not grade-related nor punitive.

Grading

For the sake of simplicity, each individual item (homework, midterm, final project) is worth 100 points. At the end of the semester, your points in each category are totaled and scaled according to the following rubric. The sum of your scaled values determines your final grade (G).

60% Homework Assignments	$G \geq 93$: A	$73 \leq G < 77$: C
15% Midterm Programming Exam	$90 \leq G < 93$: A-	$70 \leq G < 73$: C-
25% Final Creative Project	$87 \leq G < 90$: B+	$67 \leq G < 70$: D+
0% Attendance & Participation	$83 \leq G < 87$: B	$63 \leq G < 67$: D
	$80 \leq G < 83$: B-	$60 \leq G < 63$: D-
	$77 \leq G < 80$: C+	$G < 60$: F

Minor errors on homework and the midterm exam will result in small point deductions (ca. 1-4 points), while more significant errors will result in more substantial deductions (ca. 5-10 points). A generalized rubric is as follows:

100:	completely correct and error-free, code written very clearly
90-99:	mostly correct/error free, largely functional code, very few minor errors, evidence of good effort
80-89:	some minor/major mistakes, partially functional code, evidence of moderate effort
70-79:	significant errors, responses minimally correct and/or non-functional, fair/poor effort
60-69:	major issues throughout, sloppy/non-functional code, intent very unclear, minimal/no effort

The final creative project will be graded similarly, but will also take creativity into account, with a generalized rubric as follows:

100:	outstanding effort, clear evidence of creativity and critical thinking, technically flawless

90-99: very good effort, notable evidence of creativity, technically solid, few minor issues
80-89: acceptable effort, evidence of some creativity, mostly functional, some non-trivial problems
70-79: marginally adequate effort, questionable evidence of creativity, tenuous technical implementation
60-69: poor effort, little/no creativity or critical thinking, incomplete code, major technical problems

Late Assignment Submission

Grades will receive a penalty of 10% of the total assignment worth for each 24-hour period that has begun past the assignment deadline, capped at 50%. There is a grace period of up to 30 minutes past each submission deadline, and "lateness" will be determined by the timestamp on your assignment submission. In the case of late work, it is better to finish your work thoughtfully within these first 24 hours, rather than hastily throwing it together as fast as possible.

If you feel you need an extension on an assignment for any reason, you may request one via email, and it will be granted. Be mindful that work can pile up quickly if you request several extensions, so it is in your best interest to stay on top of original deadlines.

Any assignments not submitted by 11:59pm on the university-scheduled last day of instruction (Wednesday, December 9th) will receive no credit.

Course Schedule

The following schedule may undergo adjustments as needed.

WEEK	DATE	TOPIC(S)
1	Aug 27	intro to the SC environment, code evaluation, classes, instances, methods, variables, comments, getting help & other resources, the basics of making sound
2	Sept 3	unit generators (UGens), oscillators, noise generators, envelopes, basics of synth "patching," value range operations, randomness, arrays and multichannel expansion
3	Sept 10	Synth and SynthDef, arguments, outputs, iteration methods, conditional branching
4	Sept 17	filters, panners, buffers & audio samples
5	Sept 24	server architecture & the node tree: Nodes, Groups, Busses, Order-of-Execution, signal generators vs. processors
6	Oct 1	MIDI
7	Oct 8	introduction to patterns **midterm exam assigned Oct 8th 4pm CDT, due Oct 9th 4pm CDT**
8	Oct 15	pattern, continued; real-time control, TempoClock, rest events, quantization
9	Oct 22	graphical user interface design (GUI)
10	Oct 29	Open Sound Control (OSC), **final project proposals due Nov 5 by 3pm CDT**
11	Nov 5	event sequencing, composing a piece in SC
12	Nov 12	advanced synthesis topics (additive, FM, wavetable, granular, FFT)
13	Nov 19	advanced synthesis topics continued
14	Nov 26	**no lecture (fall break)**
15	Dec 3	advanced synthesis topics continued, and/or the Pen class, drawing & animation
—	TBD	presentation of final creative projects