Music 420B Administrative Info

Center for Computer Research in Music and Acoustics (CCRMA)
Department of Music, Stanford University
Stanford, California 94305

Spring Quarter, 2012-2013

Contents

Music 420B: Software for Sound Synthesis and Audio Effects

1 Course Overview

Music 420B is about preferred software embodiments for digital sound synthesis and audio effects. The principal activity is a software project due at the end of the quarter. During the quarter, the Faust language for audio signal processing is introduced, followed by a tour of plugins and synthesis/effects applications in order of increasing complexity, more or less. Familiarity with C++ and the Synthesis Tool Kit (STK),[1] or equivalent, is also assumed.

1.1 When, Where, Who

Term:	Spring Quarter
Location:	CCRMA Class Room (Knoll 217)
Time:	Tuesdays, 3:15-4:30 PM
Instructor:	Julius Smith (jos@ccrma.stanford.edu)
Website:	`https://ccrma.stanford.edu/courses/420b/`

1.2 Prerequisites

The prerequisite for Music 420B is Music 420A[2] or equivalent experience with signal processing models of acoustic systems, and software for virtual musical instruments and audio effects in C++.

Prerequisite Software

`C++` and Matlab[3] (or Octave[4]) are recommended for project work. No prior experience with the FAUST language is assumed.

2 Administrative Information

2.1 Announcements

Class announcements are often made via *email*. For this we are presently using Piazza:

`https://piazza.com/stanford/spring2013/mus420b`

You should have received an invitation from Piazza to join the class after you signed up for it in axess (using the email address known to axess). Otherwise, please join by visiting the above URL and entering your preferred email address.

[1] `http://ccrma.stanford.edu/software/stk/`
[2] `https://ccrma.stanford.edu/courses/420a/`
[3] `http://www.mathworks.com/`
[4] `http://www.octave.org/`

2.2 Units

You may sign up for 1 to 10 units. The first unit corresponds to weekly class attendance. Each remaining units should represent approximately 3 hours of focused activity per week on your project work.

2.3 Course Project

Often the 420B project is a continuation of a 420A project. Otherwise, the same kinds of project activities are available as in 420A. It is best to keep track of your time (roughly) on each category of activity. This lets you know when you've "done enough by definition," and it will help when writing your end-quarter report.

For some discussion of recommended project structure, see the Music 420A info.[5]

2.3.1 Project Schedule

The following applies to those enrolled for more than one unit:

1. Project plan due by the end of 2nd week of classes
2. Progress report due by the end of the fifth week of classes (about half way through the quarter)
3. Optional (but highly encouraged!) in-class project presentation scheduled during the last week of classes
4. Final written report due by the end of finals week

2.3.2 Project Midterm Progress Report

An efficient progress report is a rough draft of the final report including placeholder summaries of envisioned remaining work (not binding). At least the Introduction and Bibliography should be essentially complete at the mid-quarter stage. For continuations of 420A projects, one may simply update the 420A report.

2.4 Grading

Grades are based on your general activity, midterm progress report, final written report, and in-class project presentation, if any. Those enrolled for one unit (weekly meeting attendance only) should sign up for CR/NC.

2.5 Class Home Page

`https://ccrma.stanford.edu/courses/420b/`

[5] `https://ccrma.stanford.edu/~jos/intro420/Final_Project_Optional_4th.html`

3 Schedule

Below is the schedule of weekly in-class presentations, with pointers to all associated reading, lecture overheads, and so on. The default weekly topic is learning the FAUST language for signal processing. Additional topics will be driven by the interests, projects, and research activities of the participants. Results and associated pointers will be logged here.

- Week 1: General discussion of goals, and introduction to the default weekly lecture topic: Introduction to the Faust Language for Audio Signal Processing
  - Read "MUS420B Administrative Info"[6] (this document)
  - Read "Audio Signal Processing in Faust"[7] up to the section on "Term Rewriting"
  - Download and compile the FAUST compiler.
  - Briefly look over "Faust Tutorial"[8]
  - Briefly look over "Faust Quick Reference"[9]
- Week 2
  - Features of the FAUST language, following "Audio Signal Processing in Faust"[10]
  - Bowed string modeling with Esteban Maestre and Gary Scavone
- Week 3
  - Discussion of projects
  - Linear Prediction (LP) Overview: `http://ccrma.stanford.edu/~jos/LPC/`
  - Percussion modeling
- Week 4
  - Project Updates:
    - Pablo Castellanos project update: Gourd modeling
    - Kitty Shi project update: Guqin modeling
    - JOS project update: Drum modeling (using linear prediction and invfreqz in matlab)
  - Tutorial on the use of `invfreqz` and/or linear prediction to fit filters to measured data in matlab[11]
  - FAUST[12] language intro
- Week 5

[6] `https://ccrma.stanford.edu/~jos/intro420b/`
[7] `http://ccrma.stanford.edu/~jos/aspf/`
[8] `http://faudiostream.cvs.sourceforge.net/viewvc/faudiostream/faust/documentation/faust_tutorial.pdf`
[9] `http://faudiostream.cvs.sourceforge.net/viewvc/faudiostream/faust/documentation/faust-quick-reference.pdf`
[10] `http://ccrma.stanford.edu/~jos/aspf/`
[11] `https://piazza.com/class#spring2013/mus420b/11`
[12] `http://ccrma.stanford.edu/~jos/aspf/`

  - JOS project update: Synthetic bridge admittance, transmittance, and microphone measurements, based on `https://ccrma.stanford.edu/~jos/pasp/Matlab_Passive_Reflectance_Synthesis.html`
  - Myles Borin project update: Flocking and the `asm.js` Javascript subset
  - Reza Payami project update: FAUST to AU (`faust2au`)
  - Thomas Walther project update: guitar-controlled additive synthesis
  - Yan Michalevsky project update: Faust VST/VSTi plugins: working in MuLab, but not in Ableton Live

- Week 6
  - Digital State-Variable Filters[13]
    * Normalized Second-Order Continuous-Time Lowpass Filters
    * Bode Plots: Lowpass, Bandpass, Highpass, Notch variations
    * Bode Plots: Corner Resonance
    * State Space Realization
    * Digitization
    * Faust Encoding
    * Further Reading

- Week 7
  - Functional programming languages, pattern matching
  - **References**
    * Paul Hudak, "Conception, evolution, and application of functional programming languages," ACM Computing Surveys, vol. 21, pp. 359411, Sept. 1989, Available[14] without fee for noncommercial use
    * Albert Gräf, "Term rewriting extension for the FAUST programming language," Proceedings of the 8th International Linux Audio Conference (LAC-10), Utrecht, http://lac.linuxaudio.org/, 2010, `http://lac.linuxaudio.org/2010/papers/30.pdf`

- Week 8
  - Project Updates
    * Bjoern Ehrlach on acoustic bass modeling using a hybrid waveguide/mass-spring approach
  - FAUST[15] programming examples related to Music 320 and 420A
    (all in the FAUST-distribution `examples` directory)
    * Sine oscillator, cubic nonlinearity, etc., in `examples/cubic_distortion.dsp`
    * Sawtooth oscillator, flanger, phaser in `examples/phaser_flanger.dsp`

[13] `https://ccrma.stanford.edu/~jos/svf/svf.html`
[14] `https://ccrma.stanford.edu/~jos/pdf/FunctionalProgramming-p359-hudak.pdf`
[15] `https://faust.grame.fr`

    * Parametric EQ sections in `examples/parametric_eq.dsp`
    * Amp follower, gate, compressor, etc., in `examples/gate_compressor.dsp`
    * More oscillators, Moog VCF, etc., in `examples/virtual_analog_oscillators.dsp`
    * More VCFs in `examples/vcf_wah_pedals.dsp`
    * Filter bank support in `examples/graphic_eq.dsp`
    * FDN Reverberation in `examples/reverb_designer.dsp`
      (Also don't forget ZitaRev1 in `examples/zita_rev1.dsp`)
      (See also `reverb_tester.dsp`)

- Week 9
  - OSC usage in FAUST
  - A nascent FAUST drum kit
  - Week 10 - Project Presentations
    * Reza Payami: `faust2au`
    * Pablo Castellanos: Berimbau virtual acoustic modeling
    * Kitty Zhengshan Shi: Chinese Guqin virtual acoustic modeling
    * Priya Shekhar: APHEX Aural Exciter virtual analog modeling
    * Yan Michalevsky: FAUST `vsti-poly.cpp` architecture file
    * Spencer Salazar: using analog circuits for acoustic simulation
    * Thomas Walther: Guitar to additive synthesis using pitch tracking and multiband frequency-shifting
    * Tim O'Brien: Quantizing real-time FFTs in SuperCollider to achieve audio effects
    * Gina Collecchia: Estimating violin body mode radiativity from bridge signals and mic recordings
    * Myles Borin: FAUST apps in Web browsers using `asm.js`
      `https://ccrma.stanford.edu/~mborins/420b`
    * Lauchland Casey: Modal synthesis of percussion in SuperCollider