

TEACHING DIGITAL AUDIO PROGRAMMING: NOTES ON A TWO-YEAR COURSE SEQUENCE AT UCSB

Stephen Travis Pope
Graduate Program in Media Arts and Technology
University of California, Santa Barbara
stp@mat.ucsb.edu

ABSTRACT

The MAT 240 *Digital Audio Programming* course sequence is a six-quarter (i.e., two-year) practical workshop class devoted to teaching digital audio processing techniques and software development at the graduate level. It has been delivered through several complete iterations at UCSB since 2000. In this paper, we will introduce the course sequence topics, describe what students actually do and learn in the course, and evaluate our challenges, successes and failures.

1. INTRODUCTION

This paper introduces and evaluates the MAT 240 Digital Audio Programming course sequence taught in the Graduate Program in Media Arts and Technology (MAT) at the University of California, Santa Barbara (UCSB). MAT 240 [1] is a six-quarter (two-year) practical course devoted to digital audio programming and multimedia application development; it is run as a hands-on software development studio, with a mix of lectures and workshop meetings. Students read the a selection of papers from the literature, with the emphasis on learning to use and then extend the current state-of-the-art audio signal processing algorithms, programming methods, tools, and library APIs.

We will introduce the MAT 240 curriculum first, and then discuss the challenges we faced in developing and delivering such a course sequence, and finally evaluate the current syllabus and course materials.

2. TOPICAL OVERVIEW

The topics of the six quarters of MAT 240 (A-F) are:

- A: File I/O and Streaming Media
- B: Spectral Transformations
- C: Spatial and Surround Sound
- D: Sound Synthesis Techniques
- E: Control and Distributed Processing
- F: Audio Analysis and Feature Extraction

Each course lasts ten weeks and consists of a collection of materials and exercises that will be introduced below. The individual topics for the six quarters of MAT 240 are presented in the following paragraphs.

2.1. MAT 240A: Digital Audio Programming: Using I/O APIs (Fall 1)

MAT 240A is a practical introduction to modern application development tools, libraries, and interfaces for digital audio programming. We focus on (1) off-the-shelf support for sound I/O and streaming on modern operating systems, and (2) plug-in APIs for sound I/O and processing; we do not deal with signal synthesis or processing, but rather focus on I/O and distribution.

The theoretical component of the course addresses sound representation (sample rates and formats, encoding and compression) and block-oriented processing (interleaved and non-interleaved), call-back APIs and multi-threaded programming, and general multimedia software development topics.

Starting with platform-specific sound APIs such as DirectX, CoreAudio, and LADSPA, all students are expected to work on all three common operating systems and their respective C/C++ development tools. We then investigate cross-platform sound APIs such as PortAudio, LibSndFile and SndLib. The final topic is plug-in APIs such as VST, JACK and AudioUnits.

Student projects generally start with simple play and record programs, and then grow to include fancier audio players, MP3 decoders, audio-over-IP streaming and various plug-in applications. As an example, Figure 1 below shows a player developed by Ryan Avery in MAT 240A; it supports OGG Vorbis files and adds a variety of DirectX plug-in effects. Figure 2 shows Ingrid Skei's extensions to the JavaSound Groove sample player, a simple loop sequencer with MIDI IO and presets, to which she added an improved GUI and preset save/restore.

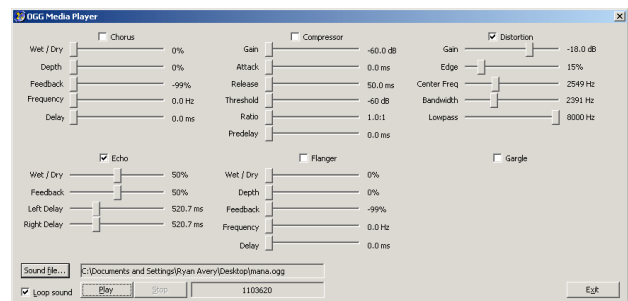


Figure 1. Ryan Avery's OGG Player with Effects

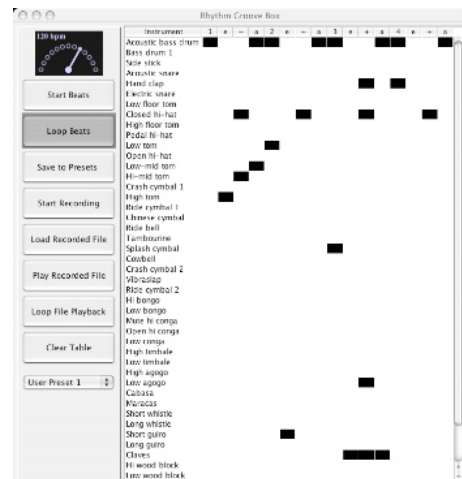


Figure 2. Ingrid Skei's Extended Groove Player

2.2. MAT 240B: Digital Audio Programming: Spectral Transformations (Winter 1)

This course focuses on the development of software for the spectral processing of digital sound. We use several libraries for spectral analysis, processing, and resynthesis (e.g., FFT libraries and vocoder programs), as well as exploring digital filter design software and other spectral effects and transformations.

We start with an overview of time- and frequency-domain signals, move then to a detailed section on the time-frequency transforms (esp. Fourier), and use several libraries for slow and fast FTs (e.g., F. R. Moore's phase vocoder and FFTW). The third topic is digital filter theory and design; we write both canonical form and FIR/IIR filters, and implement a collection of filter design techniques. This is followed by a discussion of linear prediction and LPC analysis and playback. The final topic is pitch detection methods, and we complete the quarter with group work on integrated applications

Students in MAT 240B write applications ranging from simple filter plugins to FFT-based equalizers and convolution tools to fascinating vocoder-based effects. There are also always a couple of students that work together to implement the core FFT or LPC algorithms in C themselves.

2.3. MAT 240C: Digital Audio Programming: Spatial Sound Manipulation (Spring 1)

The topic of MAT 240C is the processing of digital audio signals for performance over multi-channel output systems. Starting with simple stereophonic models, we investigate the representation of sound in space and the processing techniques for producing convincing spatialized sound. We also develop examples that use several existing surround sound APIs.

The first topics are stereo recording and processing, and multi-channel matrix encoding and playback, followed by a detailed discussion of room acoustics, reverberation and reverberator design. After a discussion of the head-related transfer function (HRTF), and its use for spatial sound playback, we move on to the three central sound spatialization techniques used in many-channel systems: vector based amplitude panning (VBAP), the Ambisonic representation and playback, and wavefield synthesis.

The development projects in this quarter start with simple panners and matrix up-mix players, and progress to time-domain and convolution-based reverberators. For the second half of the course, we work with larger-scale spatialization software for the Creatophone playback system and in the UCSB AlloSphere (see the progress report elsewhere in these *Proceedings*).

2.4. MAT 240D: Digital Audio Programming: Sound Synthesis Techniques (Fall 2)

In this quarter, we implement a variety of software sound synthesis techniques, ranging from traditional additive, subtractive, and non-linear synthesis to more contemporary physical model systems and granular and modal synthesis packages. We explore the internals of several existing synthesis packages (e.g., CSL, Csound, STK, JSyn and CLAM), and write our own versions of several popular synthesis techniques within these frameworks or as separate plug-ins or stand-alone servers.

Student projects from MAT 240D are often packaged as software synthesizers, as plug-ins for popular control and synthesis APIs (e.g., VSTi), or as Max or PD synthesis objects. Figure 3 shows the use of a PD plug-in wave-shaper developed in MAT 240D by Alex Norman, and Figure 4 below that is a visualization of a multiple pendulum sound synthesis technique developed by Will Wollcott.

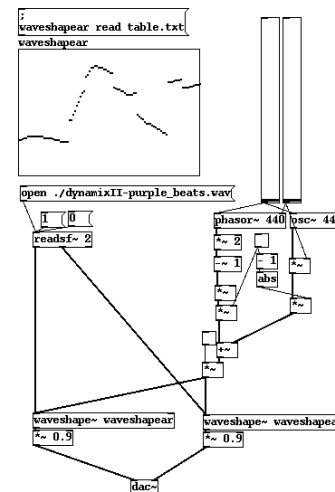


Figure 3. Using Alex Norman's PD-based waveshaper object

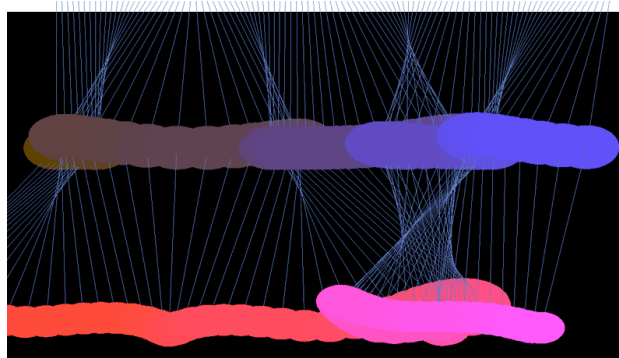


Figure 4. Visualization based on Will Wollcott's Multiple Pendulum Sound Synthesis Technique

2.5. MAT 240E: Digital Audio Programming: Multi-rate Control and Synchronization (Winter 2)

The MAT 240E course focusses on (1) protocols for real-time control of sound synthesis and processing, primarily the Musical Instrument Digital Interface (MIDI), the OpenSoundControl (OSC) protocol, and the USB human interface device (HID), and (2) simple network streaming of real-time audio using protocols such as UDP and RTP. The goal is to write programs that map in-coming data from haptic controllers, and to use this data to control networks of hardware and software synthesizers and processors. More ambitious projects will address the issues of distributed synthesis and control routing in multi-server systems.

Projects for MAT 240E often integrate projects from earlier quarters (e.g., vocoders from 240B or soft-synths from 240D) with MIDI, OSC or USB/HID interactive control. Many of the students in this class also take Dan Overholt's sensor/interface workshop [2]

2.6. MAT 240F: Digital Audio Programming: Audio Analysis and Music Information Retrieval (Spring 2)

The focus of the MAT 240F course is on audio analysis and signal processing techniques applied to sound/music databases and music information retrieval systems. We work with several different C++ libraries for signal analysis and feature extraction to develop skills in time-domain processes such as beat following, tempo

analysis, and song segmentation, and in spectral-domain analysis techniques such as pitch estimation, spectral peak analysis and tracking, and instrument signature identification. The topic of feature vector design plays an important role in the development tasks.

We use several MIR frameworks in C++, including Marsyas, Aubio, and FMAK in MAT 240E; typical student projects incorporate music segmentation, finger-printing, thumb-nailing, databases for clustering and genre classification, and user preference matching into applications such as smart players of query-by-humming systems. The Figure below shows C. Ramakrishnan's iTunes plug-in visualizer we call the "WoonGram" after MAT alumnus Woon Seung Yeo.

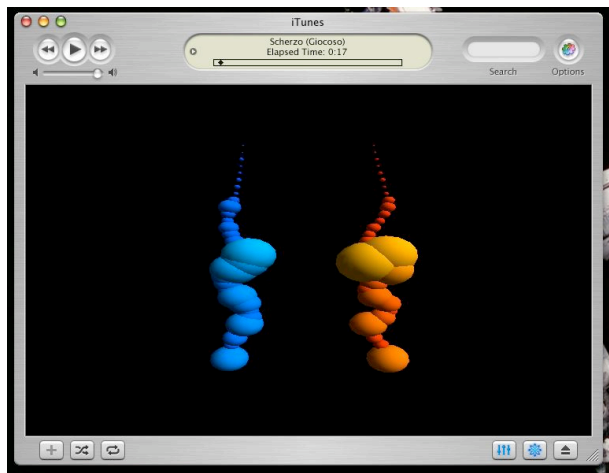


Figure 5. iTunes visualizer based on real-time feature extraction by Sekhar Ramakrishnan using Woon Yeo's feature mapping and representation.

3. COURSE MATERIALS

There are several kinds of materials associated with each course in the MAT 240 sequence; first, each course has a prepared reader (typically 200-300 pages, tables of contents available on-line), and the course web sites include web link lists for further reading. Second, students are provided with PDF files of the complete course presentation slides (also available on line). Central to each section is a large archive file of code examples for the domain at hand; these generally include open-source frameworks related to the topic and student example code from previous iterations of the course. The fourth kind of course materials are the listening selections, which range from short sound examples to full-length compositions that demonstrate a technique relevant to the particular course. The in-class presentations (40 hours per course) consist of the lecture portion (as little as necessary, up to 50%), software demonstrations, in-class debugging, and student software "show and tell."

There is a large difference between the various courses in the sequence in terms of the volume of the reader and the number of presentation slides. Generally, the earlier courses have fewer readings and presentations (focussing much more on small student projects), while the latter ones (esp. 240C and F) have both larger readers and more comprehensive lecture materials. In these courses, the students projects also

grow in scope, and are often undertaken in groups of 2-4 students.

4. PREREQUISITES, COURSE WORK

We have not yet specified the precise technical or artistic level of these courses. The MAT program is a unique interdisciplinary graduate program offered at UCSB, and MAT students come from a variety of undergraduate and post-graduate backgrounds. In the MAT 240 sequence, we generally have a mix of (a) engineering graduate students, who are expert programmers but may not be familiar with digital audio processing, (b) composers and performers who have taken electronic music courses, but may not be truly proficient C/C++ programmers, and (c) others (e.g., with visual arts backgrounds), who may have neither of the above sets of skills, but are often very eager to acquire them.

At the core of the MAT curriculum is a set of five required courses: one from each of the "host" departments (music, art, electrical engineering, and computer science) and one "catch all" course. It is important for MAT 240 that the students are expected to take the two required engineering courses in their first year. The electrical engineering core course (MAT 201A) has been described in these pages [3], and the computer science core course (MAT 201B) was designed by this author as an introduction to multimedia software development.

MAT 240 students are expected to know "the basics" of digital audio signal representation and processing, and to be "moderately" proficient in C, C++ or Java (with others such as Smalltalk, SuperCollider or scripting languages encouraged). We also make the assumption that they have used a software development environment such as VisualStudio, Xcode, KDE, or Eclipse. Programming assignments are given in C, C++ and Java, and involve small- to medium-scale software development tasks on Linux/UNIX, MS-Windows, Macintosh, audio hosts, Web browser-plug-ins, and possibly other platforms. In the early courses, students are strongly encouraged to learn the development tools and basic multimedia APIs on all three mainstream operating systems. The course work consists of reading, in-class discussion, and software development tasks, some of which are assumed to be undertaken alone, and some in groups.

5. CHALLENGES

In any teaching situation, one hopes for a narrow spread of levels of expertise on the part of the students. This is strongly *not* the case in the UCSB MAT graduate program, and MAT 240 series in particular. It is the goal of the MAT 240 curriculum to have enough detail to keep the interest of the technically literate students without frustrating the others, and it is a constant challenge to steer in-class discussions between these two extremes. We state at the outset that our goal is to transfer concepts and definitions first, tool and API fluency second, and technical (and mathematical) details as the lowest priority.

It is certainly an interesting topic to debate whether this is a good idea; one can question the utility of students who can use open-source APIs without being

able to understand all their source code (I believe it's useful; some of my colleagues do not agree). There are several areas where I believe it can be important to instill confidence in students (such as being unafraid to learn new platforms and development tools), even if there are cases where this confidence might be dangerous (using an API whose inner workings one doesn't understand).

Another topic of discussion is the list of topics, their order, and their relative weighting. Some students take most of the courses in the MAT 240 sequence, but skip one or the other quarter because they feel that its topic is not central to their interest (or not on the critical path to their thesis project). For each of the six topics of MAT 240, one could (and students do) debate the weighting of theory and practice, and of specific technologies (e.g., gaming APIs, telecomm. protocols) and their implementations (MATLAB, Max) related to the main theme of the course.

Practically and logistically, it would be much easier if we were to select a single language and development environment for the entire course sequence. Indeed, one could certainly teach the entire course on a single platform, although that would limit the number of available packages one could incorporate. As it is, students often refer to the MAT 240A course as "IDE Hell" meaning that it can be a challenge for students to move between the integrated development environments (IDEs) of the various mainstream computer platforms.

6. EVALUATION

The MAT 240 course sequence is currently in its fourth round, and the course materials have generally been revised and updated each time they are presented. Nevertheless, this is a broad curriculum in a rapidly changing area, and there are constant discussion among the MAT faculty, my teaching assistants, and course participants about what topics should be added or removed (or simply rewritten).

There are always requests to speed up the presentation and to go deeper into the individual areas, and to slow down and spend more time on the basics. As stated above, it is the working assumption that the goal is that these two poles be equally represented in the student population in any given course.

Students who have graduated from the MAT degree program with MS degrees in Media Engineering have gone on to development jobs in a whole list of well-respected digital audio companies, to small start-up companies, to careers as independent consultants, and to PhD programs at several universities.

The students for whom the MAT 240 model does not work are frequently either too novice as programmers (i.e., not facile enough with software development tools and learning new APIs), or lack the desire to learn the math details and write their own low-level C/C++ implementations of the literature (i.e., they would rather use MATLAB, Max/PD, SuperCollider or Csound).

The course materials could be prepared in a number of different (and livelier) ways, including wikis and on-line community tools for the students, shared code projects, and more interactive presentations, using, e.g., more of Bob Sturm's SSUM [3]. We have accumulated a series of student-developed programs for illustrating topics such as pole/zero diagrams and filter amplitude/phase

transfer functions, granular synthesis, vocoders, equal-power panning, various spatial distance cues, and MIDI/OSC tracing tools.

As we mentioned above, it would be easier in many respects to select a single language and platform to use for the entire course sequence (e.g., Java, JMF and Eclipse, or SuperCollider, or Siren and CSL, or Python, or MATLAB...), but the decision was made to strongly encourage students to branch out, to learn new languages and new IDEs as a good preparation for the "real world" of the multimedia software development industry. I have taught similar topics in the past using Csound, SuperCollider, Smalltalk/Siren, and other platforms, but the current MAT 240 series is oriented towards giving the students both theoretical insights and concrete job skills in today's market, so we stress platform- and language-agnosticism as a general policy.

Finally, the MAT 240 sequence has no specific explicit goal or "final exam" that would guarantee a certain common set of skills to be expected from its graduates. The kernel skills that the program wants to transfer to the students are to be accustomed to learning new models and APIs for multimedia programming, and to look for opportunities to integrate resources from several sources into new tools and instruments, creating custom components as required. The MAT graduate program is project-centric, and we expect students in any one of our concentration areas to integrate skills and tools from a variety of areas into research or creative efforts that produce tangible and novel results.

7. RESOURCES

The materials for the MAT 240 course sequence are mostly available on the CREATE and MAT web sites; see <http://create.ucsb.edu/240>. Most of the readers are not available in machine-readable form (the tables of contents are).

8. CONCLUSIONS

The MAT240 Digital Audio Programming course sequence has been delivered at UCSB since 2000, and has evolved significantly since its inception. We believe there is a need for such specialized training at the graduate level, and have seen our students proceed on to advanced positions in digital audio research and development. We would welcome any sort of collaboration with other centers wanting to use these materials for short courses, multimedia presentations and courseware, intensive workshops, or standard multi-semester course sequences.

9. REFERENCES

- [1] Pope, S. T. MAT 240A-F Web Site.
<http://create.ucsb.edu/240>. 2000-07.
- [2] Overholt, 2006. "Musical Interaction Design with the CREATE USB Interface Teaching HCI with CUIs instead of GUIs." *Proc. 2006 ICMC*
- [3] Sturm, B. L. and J. Gibson, "Signals and Systems Using MATLAB: An Effective Application for Exploring and Teaching Media Signal Processing," *Proc. 2004 ICMC*.