**Music 47 - Music, Mathematics and Programming**

# Syllabus

## Introduction

The relationship between music and mathematics has been investigated since ancient times by scholars situated around the world. This exploration has allowed us to create instruments, define musical parameters, and develop standardized systems (such as tuning systems) that aid music creation. In the 21st century, music creation has become intricately linked to computer technology adding a layer of complexity to the already complex relationship between music and mathematics. For most users of audio technology, understanding of mathematics is optional. However, for those who intend to create computer programs for musical purposes, it is a necessity. The good news is that it does not take much more than the knowledge of high school mathematics to get started in this field.

This course is designed to help those who want to create music programs and need some guidance to get started. Students will –

- develop an understanding of musical concepts through the understanding of the mathematical concepts underlying them,
- use this understanding to write programs that can be used to make music, and
- appreciate the role of mathematical concepts in electronic music production.

The objective is to introduce students to concepts that will help them begin their audio programming journey. The course will expose students to a range of topics where music and mathematics overlap. Students are expected to research those that interest them in addition to reviewing the assigned study material, and implement in their programs what they learn. They will have the opportunity to share their work during one of the weekly meetings where they will receive peer feedback. Collaboration is encouraged. It is likely that members of the class will have varying degrees of knowledge in pertinent areas. Students should actively participate in class discussion and on the Discord server with the spirit of sharing expertise and learning from others. The final project (see Course Requirements) is expected to be completed in groups of two or three.

## Topics

Most of the topics have been divided into three categories: rhythm, pitch and timbre.

### Rhythm

- Time
- Meter
- Temporal units and their divisions (bars and note values)
- Tempo
- Beats (combination of different rhythmic patterns played on different instruments)
- Sequencing

### Pitch

- Frequency
- Vibrating strings: length, tension and frequency
- Sine and cosine functions
- Relationship between pitch and frequency
- Just intonation and equal temperament tunings
- Relationship between MIDI note number and frequency

### Timbre

- The harmonic series
- Time domain vs. frequency domain
- Amplitude modulation and frequency modulation
- Envelope
- Delay and effects created with delay
- Filters
- Distortion

Throughout the course, we will keep coming back to the following topics:

- Arithmetic vs. geometric progression
- MIDI and MIDI controllers
- Control signals
- Conditional branching
- Automation
- Algorithmic music
- Digital audio basics (sampling, analog to digital and digital to analog conversion, audio interfaces, etc.)

## Software Requirements

The programming language we will use for this class is MAX, a node-based coding platform that was developed with artists in mind. It is different from text-based languages like C++ and Java in that the building blocks—known as 'objects'—are presented visually, can be moved around in the 'patcher' space, and can be connected to each other using 'patch cords'. Cycling '74 offers a 30-day free trial of MAX. Students can access the full version of MAX using the CTSA license. Additionally, students may want to download Audacity. It is free and a very useful tool for easily editing audio files.

## Activities

- Lectures on topics at the intersection of music and mathematics
- Demonstrations of music-mathematical concepts and programming applications using MAX
- Discussion sessions for students to share their programming work and get peer feedback
- Quizzes for students to demonstrate their understanding of discussed topics
- Short programming assignments related to topics discussed in class
- Presentation of a substantial programming assignment for the final project; to be completed

in small groups and submitted along with an approximately 2000-word paper about how the project was accomplished

## Course Requirements

An attendance record of 90% or better is required to pass the class. Please make every effort to arrive on time. Arrival more than 20 minutes late will be considered 1/2 absence; arrival more than 40 minutes late will be considered a full absence.

Students should attend each class session having done the assigned work—readings, viewings, research, and preparation of presentations—and be prepared to participate actively in discussion.

Students will be responsible for demonstrating their completion of the assigned studies by successfully answering a short weekly quiz.

Students will complete some short programming assignments, which will be evaluated on program correctness and functionality, thorough fulfillment of the stated requirements, and demonstrated effort. The assignment should be submitted before the Tuesday session every week. On Thursday, the class will discuss and give feedback on as many submissions as time permits.

At the end of the quarter, students will present their final project which will be a substantial programming assignment that they will work on in groups of two or three. Students will have the option to either complete a defined task with the tools of their choosing, or propose a different project of similar magnitude. Students opting for the second option are required to submit their proposal by the end of week 5. Upon instructor's approval, they will find collaborators. Each group should also submit a short paper (approximately 2000 words) outlining how they approached the task in terms of design, programming and research.

## Textbooks

The class will be asked to read chapters from *Musimathics, Volume 1: The Mathematical Foundations of Music* by Gareth Loy. The book can be bought from The Hill Bookstore. They do offer price matching. We will use several online resources as well, such as *Max Cookbook* and *Computer Music Programming* by Christopher Dobrian et al.

## Grading

Students will be graded on their performance in quizzes, short programming assignments, and a final project consisting of a substantial programming assignment, a presentation and an accompanying paper. They will be weighted as follows:

- Quizzes - 20%
- Short programming assignments - 40%
- Final project - 40%
    - Program - 20%

- Presentation - 10%
- Paper - 10%

The quizzes will feature questions about topics discussed in the lectures. Though the number of questions in the quizzes may vary, the scores will always be normalized to a range of 0 to 10. For example, if a quiz has 7 questions and a student gets 6 of them right, they will receive 8.6 points.

The short programming assignments will be graded in the following way. A student gets 2 points if their program meets the criteria outlined by the assignment, 1 if all the criteria are not met, and 0 if the program does not work at all or if nothing was submitted. If a student goes beyond the instructions and creates a more sophisticated (and, of course, working) program, they will receive an extra point resulting in a 3. The extra points will compensate for points lost elsewhere.

The final project will be a group assignment. A group's score will reflect on all individuals in the group. The program will be assessed based on how well it accomplishes its proposed task, elegance of the code, and demonstrated effort. In case of the presentation, clarity, conciseness, and preparedness will be considered. The assessment of the paper will take into account clarity, attention to details, and how well the specific information asked for have been incorporated.

## Communication

Students are encouraged to reach out to the instructor with any questions regarding the course through UCI email or Canvas. Discussion outside of class will take place primarily on the course's Discord server.

## Course Summary

Lectures will take place on Tuesdays and discussions on Thursdays. Quizzes and assignments should be completed by Tuesday the following week.

### Week 1

- Lecture: Ice breaker, review of arithmetic and geometric progressions, carrying out mathematical operations in MAX, and data types
- Discussion: Setting up a coding environment
- Quiz: Algebra basics
- Programming assignment: Make a calculator that performs addition, subtraction, multiplication and division between two numbers

### Week 2

- Lecture: Pitch, scales, tuning systems, representing pitches as numbers, and how to make an oscillator play the chromatic scale
- Discussion of last week's programming assignment
- Quiz: Pitch, frequency, and MIDI note numbers
- Programming assignment: Control the pitch of an oscillator with a computer keyboard

**Week 3**

- Lecture: Time, performing tasks at specific intervals, rhythm, and envelopes
- Discussion of last week's programming assignment
- Quiz: Rhythm
- Programming assignment: Modify last week's assignment patch to trigger different pitches and envelopes at set intervals without user input

**Week 4**

- Lecture: Introduction to algorithms (what they are and what they do), Euclid's algorithm and how it can be used to create rhythmic patterns
- Discussion of last week's programming assignment
- Quiz: Algorithms
- Programming assignment: Modify last week's assignment patch to make two or more overlapping Euclidean patterns each playing a fixed but distinct pitch

**Week 5**

- Lecture: Time domain vs. frequency domain, harmonic series, and filters
- Discussion of last week's programming assignment and using MIDI controllers
- Quiz: Harmonic series
- Programming assignment: Make a monophonic subtractive synthesizer that can be played with a MIDI keyboard controller, is velocity sensitive, uses a complex waveform, and gives user a choice of two different kinds of filter
- For students who want to work on a final project different from the one assigned - proposals for final project due

**Week 6**

- Lecture: Modulating amplitude, frequency and other parameters, and delay
- Discussion of last week's programming assignment
- Quiz: Modulation
- Programming assignment: Modify last week's assignment patch to add a second oscillator that modulates the frequency and/or amplitude of the first
- Assigning students to groups

**Week 7**

- Lecture: Randomness, chance/probability, convergence, divergence, etc.
- Discussion of last week's programming assignment and final project
- Quiz: Probability
- Programming assignment: Modify last week's assignment patch so it plays notes on its own without user input, but the pitches and how often they are played should result from a probability function

**Week 8**

- Lecture: Algorithmic composition - what it is and why do it

- Discussion of last week's programming assignment and final project
- Assignment: Submit a plan for how the group will approach the final project

**Week 9**
- Lecture: Effects made with delay - flanger, phaser, chorus, etc.
- Discussion of final project
- Programming assignment: Work on the final project and submit a short progress report

**Week 10**
- Lecture: Dynamics processing (compressor, limiter, noise gate and expander) and distortion
- Presentation of final projects

**Finals Week**
- Presentation of final projects
- Final project submission

## Disability

If you have a disability that inhibits you from performing any of the stated requirements of this course, as approved and documented by the UCI Disability Services Center, please ensure that the professor is thoroughly aware of the matter as early in the term as possible.

## Academic Honesty

Collaboration between students in this course is strongly encouraged. Students are urged to exchange ideas, opinions, and information constantly, and to help each other with the research and programming projects. However, each student is responsible for completion of their own assignments.

Plagiarism of any kind is in direct violation of the UCI policy on Academic Integrity, and penalties for plagiarism can be severe. In this class you will be expected to attribute due credit to the originator of any ideas, words, programming code, or other ideas that you incorporate into your own work. Any borrowed text must be cited in proper academic bibliographic fashion, giving credit to its original author. Any borrowed computer programming code must be cited in inline commentary in the code, and in any documentation written about the code, giving credit to its original author.

In computer programming, it's common to use program components that are known to be reliable, written by others. A lot of good (and some bad) program code is freely available. Nevertheless, one must always give full attribution to the original author of all program code.