**Computer Music on a Laptop: Composing, Performing, Interacting**
(3 hours, sophomore and up, no prerequisites, co-taught by two Music and Computer Science faculty)

**Professor:**
Dr. Bill Manaris

**Office:**
Room: 223 J.C. Long Building
Phone: (95)3-8159
E-mail: manarisb@cofc.edu

**Professor:**
Dr. Yiorgos Vassilandonakis

**Office:**
Room: 204 Cato Center for the Arts
Phone: (95)3-3845
E-mail: vassilandonakisy@cofc.edu

**Catalog Course Description**

Principles of music composition and computer programming for developing interactive computer music environments. Team-based, project-driven exploration of Python programming, time-based structures, algorithmic processes, soundscapes, graphical user interfaces, musical language and style.

**Prerequisites by Topic**

1. Basic computer experience, including file organization and software installation.
2. Interest in developing intelligent-listening and sound-structuring skills.

**Additional Course Requirement**

Students need their own laptops and headphones.

**Outline of Course Topics**

*Computer Science:* Algorithm, setting up Python and jMusic, data types, variables, assignment, arithmetic operators, I/O, MIDI language and protocol, selection (if statements), relational operators, iteration (loops), lists, functions, audio representations (WAVE, MP3), modularization (functions), classes (object-oriented design), events, and graphical user interfaces (GUIs).

*Music:* Aural awareness, musical space (line, register, field), sound elements (envelope, timbre) and properties (pitch, rhythm, timbre, spatial orientation), time structuring & time scales (rhythm and pulse
complexes, event density, beat modulation, serialization, palindromes, open time fields, psycho-physical
time), pitch organization systems, (horizontal and vertical), grammar and syntax of traditional and
non-traditional musical languages (communications systems), gesture-form-structure, global musical
structures and systems, applications of fractals in musical time and space, soundscape & texture building,
interactive music models, psychoacoustics, basic principles of composition.

Synthesis Experiences: Students will follow pair-programming paradigm to develop music compositions
through algorithmic design and implementation in Python.

Learning Objectives

- Introduce music composition in the context of designing sound in time, graphic representation
  (musical notation), sound as a building block, basic principles of sound organization.
- Learn basic principles for group collaboration.
- Apply Python numeric and string data types to represent information.
- Use variables in program development.
- Understand arithmetic operators and use them to design expressions.
- Understand sound as a time-based event with spatial parameters.
- Create representations of notes, phrases, parts, and scores in Python.
- Create divisions of space and time continua into small increments and organize them with scales.
- Understand the concept of modular design in sound.
- Understand the basic algorithmic building blocks: sequence, selection, iteration, and modularization
  (functions and objects).
- Apply additive processes to pitch and rhythm cells.
- Understand the concept of indeterminacy within certain levels of divisions of musical space, while
  maintaining determinacy in large-scale structure.
- Understand Python’s random operations and apply (map) them to create musical structures.
- Analyze methods of organizing global musical structures.
- Understand Python’s if statement and apply it to implement algorithmic selection patterns.
- Understand Python’s for-loop statement and apply it to implement algorithmic iteration patterns.
- Understand sound blocks and sound masses.
- Apply Python’s list data structure to represent sequences of data.
- Understand Python’s functions and apply them to modularize algorithmic processes.
- Develop musical gestures and incorporate them within global formal designs.
- Understand Python’s classes and apply them to build computer music instruments.
- Analyze and manipulate musical timbre.
- Understand and apply principles of event-driven programming.
- Learn graphical user interface (GUI) widgets and use them to create basic GUIs.
- Understand basic principles of psychoacoustics and music perception and cognition.
- Analyze existing idioms and generate a new one.
- Create interactive computer music instruments.
Grading

Scale: A: 90-100; B: 80-89; C: 70-79; D: 60-69; F: <60. The grades of B+/-, C+/-, and D+/- may be given at the professor's discretion.

Final Grade Computation: Projects (1-4) 60%, Final Project 20%, Listening and Reading Journal: 10%, and Class Participation 10%

Readings

Required texts

- Brown, Andrew and Bill Manaris. Making Music with Python, draft manuscript.

Supplementary readings


**Supplementary audiovisual materials**

● *Steve Reich: A New Musical Language*. [http://www.youtube.com/watch?v=3xON0AAYdVw](http://www.youtube.com/watch?v=3xON0AAYdVw)

**Course Schedule (tentative)**

**Unit 1: Introduction and Concepts**

<table>
<thead>
<tr>
<th>Week 1</th>
<th>Music</th>
<th>Computer Science</th>
</tr>
</thead>
<tbody>
<tr>
<td>F 8/26</td>
<td>20th century innovations in music conception.</td>
<td>Why make music with computers (data, algorithm, sequence, selection, repetition, randomness, modularization, interaction, algorithmic music composition).</td>
</tr>
</tbody>
</table>

**Readings:**

Michael Edwards: *Algorithmic* The computer as a musical instrument.
Computation: Computational Thinking in Music.

Iannis Xenakis: *Formalized Music* Chapter X: Concerning Time, Space and Music

Edgar Varese: *Open rather than bounded.*

<table>
<thead>
<tr>
<th>Week 2</th>
<th>Music</th>
<th>Computer Science</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>time, from antiquity to the present.</td>
<td>Operating system basics.</td>
</tr>
<tr>
<td>W 8/31</td>
<td></td>
<td>Creating your first Python program (first.py).</td>
</tr>
<tr>
<td>F 9/2</td>
<td></td>
<td>Computer representation of music.</td>
</tr>
</tbody>
</table>

**Viewing:**

Seth Horvitz – Eight Studies for Automatic Piano.

**Viewing:** “There’s no sound in my head” Documentary on Mark Applebaum’s *Metaphysics of Notation.*

<table>
<thead>
<tr>
<th>Week 3</th>
<th>Music</th>
<th>Computer Science</th>
</tr>
</thead>
<tbody>
<tr>
<td>M 9/5</td>
<td>Algorithmic Thinking in Composition: From Renaissance to Xenakis.</td>
<td>Data.</td>
</tr>
<tr>
<td>W 9/7</td>
<td>Writing music in code; Python numbers, data types, variables; syntax errors; the jMusic data structure: Note, Phrase, Part, and Score.</td>
<td></td>
</tr>
<tr>
<td>F 9/9</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Project:** Transcribe a musical piece to Python and jMusic.

### Unit 2: Minimalism and MIDI

<table>
<thead>
<tr>
<th>Week 4</th>
<th>Music</th>
<th>Computer Science</th>
</tr>
</thead>
<tbody>
<tr>
<td>M 9/12</td>
<td>The “Cell”: Modular design and economy of material from Beethoven to Stravinsky.</td>
<td>Sequence.</td>
</tr>
<tr>
<td>W 9/14</td>
<td></td>
<td>Creating melody and rhythm; the jMusic data structure: Note, Phrase, Part, and Score (cont’d); repetition and phasing; Python lists.</td>
</tr>
<tr>
<td>F 9/16</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Week 5</th>
<th>Music</th>
<th>Computer Science</th>
</tr>
</thead>
<tbody>
<tr>
<td>M 9/19</td>
<td>Minimalism, Process and Phase.</td>
<td></td>
</tr>
<tr>
<td>W 9/21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F 9/23</td>
<td><strong>Viewing:</strong></td>
<td>Steve Reich Documentary</td>
</tr>
</tbody>
</table>

**Project:** Create a minimalist piece in Python and jMusic.

### Unit 3: Aleatoric Thinking in Music and Computer Science

<table>
<thead>
<tr>
<th>Week 6</th>
<th>Music</th>
<th>Computer Science</th>
</tr>
</thead>
<tbody>
<tr>
<td>W 9/28</td>
<td></td>
<td>Exploring the role of chance, uncertainty and improvisation in music making; Python random functions (rand, randint choice).</td>
</tr>
<tr>
<td>F 9/30</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Readings:**

Joel Chadabe: *Electric Sound*, Chapter 10: *Automata*

Stephen Stucky: *Lutoslawski and his Music* (excerpts)

### Week 7

**Music**

- **M** 10/3

**Computer Science**

- Modularization.

- **W** 10/5
  - Stochastic Music

- **F** 10/7
  - **Week 8**
  - Stochastic Music (cont’d)

### Week 8

**Music**

- **M** 10/10
  - Stochastic Music (cont’d)

**Computer Science**

- Selection and Repetition.

- **W** 10/12

- **F** 10/14
  - **Readings:**

  Xenakis: Stochastic Music

**Project:** Create an aleatoric piece with high-level modular design (functions) and low-level indeterminant components (randomness, if statements and loops).

### Unit 4: Timbre & Synthesis

**Week 9**

**Music**

- **M** 10/17 Fall Break
  - Introduction to Psychoacoustics and Perception and Cognition Principles.

**Computer Science**

- Computer as musical instrument.

- **W** 10/19

- **F** 10/21
  - jMusic synthesized instruments; explore a sample synthesizer consisting of 12 sinewaves (using sliders to control frequencies and amplitudes); use this synthesizer as a tool to explore evolution of timbral
Week 10 | Music | Computer Science
---|---|---
M 10/24 | Musical Timbre | 
W 10/26 | 
F 10/28 | Additive Synthesis |

**Project:** Using jMusic Sinewave instruments, design and implement a musical piece based on evolution of timbre over time.

**Readings:**
Robert Cogan: *Sonic Design*, Chapter 4: *The Color of Sound*

Week 11 | Music | Computer Science
---|---|---
M 10/31 | Spectral Music | 
W 11/2 | 
F 11/4 | 

**Readings:**
Joel Chadabe: *Electric Sound*, Chapter 6: *Synthesizers*

**Readings:**
Joshua Fineberg: *Introduction to Spectral Music*

Composing with timbre and timbral processing.

**Readings:** IRCAM: Le timbre; selected articles

Unit 5: Interactive Music
### Week 12

<table>
<thead>
<tr>
<th>Day</th>
<th>Date</th>
<th>Music</th>
<th>Computer Science</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>11/7</td>
<td>Modes of Musical Interactivity: from Chamber music to Computers.</td>
<td>Graphical user interfaces and classes (cont’d).</td>
</tr>
<tr>
<td>W</td>
<td>11/9</td>
<td></td>
<td>Development of interactive music instruments.</td>
</tr>
<tr>
<td>F</td>
<td>11/11</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Final Project assigned:**
Conceive, design, program and realize a performance of a new, original 5-7 minute composition that builds upon all the knowledge acquired during the semester.

### Week 13

<table>
<thead>
<tr>
<th>Day</th>
<th>Date</th>
<th>Music</th>
<th>Computer Science</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>11/14</td>
<td>Review material.</td>
<td>Review material.</td>
</tr>
<tr>
<td>W</td>
<td>11/16</td>
<td>Guided work on final project. Roundtable design sessions.</td>
<td>Guided work on final project. Roundtable design sessions.</td>
</tr>
<tr>
<td>F</td>
<td>11/18</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Weeks 14-15</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>11/21</td>
<td>Guided work on final project.</td>
<td>Guided work on final project.</td>
</tr>
<tr>
<td>W</td>
<td>11/23</td>
<td>Thanksgiving</td>
<td>Rehearse for final performance.</td>
</tr>
<tr>
<td>M</td>
<td>11/28</td>
<td></td>
<td></td>
</tr>
<tr>
<td>W</td>
<td>11/30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>12/2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>12/5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Projects**

**Project 1:** Music Transcription into Python and jMusic (Week 1-3, 15% of Course Grade)
Transcribe a given musical piece from a high-level notation to Python and jMusic. The goal is to learn how to represent music in different systems.

**Learning Objectives:**

- Understand how sounds events can be broken down and mapped onto grids.
- Translate from one music representation to another.
- Understand how numbers and other data types are represented and manipulated in Python.
- Create representations of notes, phrases, parts, and scores in Python.

**Requirements:**

1. Working with a partner, ensure that Python and jMusic is properly installed on your laptops.
2. Translate provided piece into Python and jMusic notation.
3. Enter your Python program using an editor.
4. Include header documentation in your program with your names, assignment name, date, etc.
5. Execute your program to ensure it generates the music correctly.

**Submit:**

1. Python program file (`nameOfPiece.py`) and MIDI file (`nameOfPiece.mid`), where `nameOfPiece` is the provided piece to be transcribed.

**Project 2: Minimalist Music and MIDI (Week 4-5, 15% of Course Grade)** Design a modular piece with limited timbre, motivic material (pitch and rhythm cells), and implement it with MIDI in Python. The goal is to learn how to work with limited limited resources by maximizing combinations.

**Learning Objectives:**

- Understand how numbers and other data types are represented and manipulated in Python.
- Create representations of notes, phrases, parts, and scores in Python.
- Apply Python’s list data structure to represent sequences of data.
- Create divisions of space and time continua into small increments and organize them with scales.
- Understand the concept of modular design in sound.
- Apply additive processes to pitch and rhythm cells.

**Requirements:**

1. Working with a partner, select pitch set and rhythmic modules as basic material.
2. Experiment with different combinations of basic material.
3. Plan global formal narrative of the piece.
4. Implement piece in Python and jMusic using only one timbre (e.g., Piano).
5. Create MIDI file of the piece.

Submit:

1. Pre-compositional / planning sketches on graph paper.
2. Python program file (nameOfComposition.py) and MIDI file (nameOfComposition.mid), where nameOfComposition is your composition’s name.

Project 3: Aleatoric Music (Week 6-8, 15% of Course Grade) Design and implement an aleatoric piece in MIDI, using controlled levels of randomness on the surface level, within a well-designed formal structure. The goal is to learn how to conceive and maintain control of high-level formal design while allowing low-level indeterminacy.

Learning Objectives:

- Understand the concept of indeterminacy within certain hierarchical levels of musical space, while maintaining determinacy in large-scale structure.
- Understand Python’s random operations (functions) and apply (map) them to create musical structures.
- Analyze methods of organizing global musical structures.
- Understand Python’s functions and apply them to modularize algorithmic processes.
- Understand Python’s if statement and apply it to implement algorithmic selection patterns in code.
- Understand Python’s for-loop statement and apply it to implement algorithmic iteration patterns in code.

Requirements:

1. Working with a partner, design a high-level formal structure for a musical piece.
2. Capture the high-level design onto graph paper.
3. Define the aleatoric parameters for each of the piece’s modules.
4. Experiment with different combinations of basic material.
5. Implement piece in Python and jMusic using only one timbre (e.g., Piano).
6. Create MIDI file of the piece.

Submit:

1. Pre-compositional / planning sketches on graph paper.
2. Python program file (`nameOfComposition.py`) and MIDI file (`nameOfComposition.mid`), where `nameOfComposition` is your composition’s name.

**Project 4: Timbral Music (weeks 9-11, 15% of Course Grade):** Using jMusic Sinewave instruments, design and implement a musical piece based on evolution of timbre over time.

**Learning Objectives**

- Understand basic principles of psychoacoustics and music perception and cognition
- Develop timbrally-conceived gestures and musical structures and incorporate them within global formal designs.
- Understand and analyze musical timbre within the context of a composition.
- Apply jMusic Sinewave instruments to explore interesting timbral structures.
- Understand the principles of event-driven programming and graphical user interfaces (GUIs).

**Requirements:**

1. Explore timbral structure using sample code (provided in class)
2. Working with a partner, design a high-level formal structure for a soundscape piece
3. Capture the high-level design onto graph paper.
4. Implement piece in Python and jMusic.
5. Create audio file of the piece.

**Submit:**

1. Pre-compositional / planning sketches on graph paper.
2. Python program file (`nameOfComposition.py`) and audio file (`nameOfComposition.aiff`), where `nameOfComposition` is your composition’s name.

**Final Project: Algorithmic Music (Weeks 12-15, 20% of Course Grade):** Working in teams of 2 to 4 people, conceive, design, program and realize a performance of a new, original 5-7 minute composition for laptop instruments. This project synthesizes the following concepts:

- jMusic data structures, sequence, if statements, loops, functions, randomness, GUIs, classes
- global musical structure plan
- process-based or minimalist rhythmic layer (optional)
- aleatoric sections with controlled improvisation layers (randomness)
- timbrally evolving sections (jMusic synthesized instruments)
- interactive component (GUIs)

The final project builds upon all the knowledge acquired during the semester, now employed towards achieving a well-defined and aesthetically-clear algorithmic compositional goal. The progress of the final project will be organized in 3 phases:
Phase 1: conceptualize, design, present to class (for feedback)
Phase 2: refine design, implement (program), test.
Phase 3: rehearse, perform, submit.

Learning Objectives

- Analyze and manipulate musical timbre.
- Apply jMusic data structures, sequence, if statements, loops, functions, randomness, GUIs, classes to synthesize a computer music system.
- Understand and apply principles of event-driven programming.
- Learn graphical user interface (GUI) widgets and use them to create basic GUIs.
- Create an interactive computer music instrument.
- Understand basic principles of psychoacoustics and music perception and cognition.
- Analyze existing idioms and generate a new one.

Requirements

1. Re-visit all prior course projects, and select the most aesthetically and technically interesting aspects.
2. Work with a partner to conceive the large plan for a new composition, following guidelines and enforcing all constraints.
3. Present your high-level concept and design to class, receive feedback, and refine accordingly.
4. Using Python and jMusic, implement the necessary material and create the interactive elements to realize the composition.
5. Test everything before going to rehearsal.
6. In group rehearsals, practice executing these patterns, controlling various parameters (e.g., number of iterations, instrumentation, and intensity) via the created computer instrument(s).

Submit:

1. Pre-compositional / planning sketches on graph paper.
2. A graphic score for laptop performers to follow during rehearsals and performance.
3. Python program files (use meaningful names).

Additional requirements will be provided based on the nature of the individual composition and what was covered during the semester.

Listening/Reading Journal (10% of Course Grade)

Students are required to keep an informal journal/log of their thoughts/Reflections/comments on the assigned reading, and listening/viewing during the semester. These will be collected and returned periodically throughout the semester.