CS 150: Algorithmic Music Composition
Final Project

Richard Townsend, Tufts University

Due Date for Code, Presentation Slides: April 22nd, 2025 at 11:59pm
Due Date for Critique: April 28th, 2025 at 11:59pm

Background

Your final project is essentially a supercharged composition lab: you will write a music21 program (or collection of programs) that uses algorithmically interesting approaches to generate a composition. You MUST work in a group of 2-4 people (no solo projects) and I highly recommend 3-4 both so you can have more interesting projects and so if someone flakes you aren't stuck working on your own.

Here are some potential "characters" of final projects (many other possibilities exist):

1. *The Researcher*: follow Richard's path for developing the course by searching for one or more research papers presenting algorithmic approaches to music composition, and implement those approaches yourselves. These papers would need to be "new", in that we haven't already covered them in class. Here are some of the sources for paper used in this course: Computer Music Journal; Journal of New Music Research; International Society for Music Information Retrieval; ACM SIGPLAN International Workshop on Functional Art, Music, Modelling and Design (FARM).
2. *The Genre Junky*: you are an aficionado of a certain composer, band, or style of music, and want to investigate how to get a program to generate a piece that "sounds like" existing works from that realm. What are the hallmarks that distinguish piano pieces written by Mozart vs Beethoven? What makes a piece of music sound like Ska? Will you finally give the people a new Beatles album? Your project will generate a new work that tries to pass the "Genre Turing Test": could an expert in that genre discern that your piece was made by a computer as opposed to the original composer/band?
3. *The CS Purist*: you're aware of some cool models or algorithms from non-musical realms of CS, and re-tool those concepts to "solve" the "problem" of composing a piece of music.

The Code

For your final project submission, you will submit the code needed to generate your composition along with a `README` file. All submitted composition programs must run in at least two modes: one mode to purely generate sheet music, and another mode to purely generate a midi (just like the composition labs). Your code should be documented enough that it's obvious to the reader where the "algorithmically interesting" compositional approaches are hiding. As with lab 2, if your program trains a model or genetic algorithm, it must be able to run in the two modes specified in the lab 2 spec.

Here are the requirements for your `README` file:

1. List the names and UTLNs of everyone in your final project group. Specifically call out the UTLN of the group member who submitted your code (so Richard can easily find it).
2. Clear instructions on how to run your submitted code. These instructions must at least cover how to generate sheet music and, separately, how to generate a midi. If you had to install any new libraries to get your code to work, those libraries should also be listed here with instructions of how to install them. If Richard cannot install these libraries, he can't run and grade your code; it's in your best interest **to check that Richard can install these libraries well before the submission deadline.**
3. A paragraph giving a high-level summary of your overall compositional approach.
4. (Optional) any additional interesting details you'd like to share about your approach or how you implemented it.

The Presentation

Each group will give a presentation on their project; Richard will specify how long these presentations must be based on the total number of groups. You should prepare a slide deck that answers the following questions:

1. What are the algorithmic approaches that produced your composition?
2. Where are the algorithms "hiding" in the generated music? Show snippets of sheet music and other visualizations!
3. What would you do differently if you had to do this again?
4. What suggestions would you have for future students designing their final composition projects?

Between the slides answering questions 1 and 2, you will run your code, live, to "play" your composition. You have four options for playing the whole piece (but the last two options must be preceded by a live run of the program, which you can then cut off after a few seconds of play time):

- Run the program, live, to generate a midi that plays.
- Run the program, live, to generate sheet music that is then played by the sheet music software.
- Record a video of yourselves performing the piece and insert it in a slide.
- Perform the piece yourselves, live.

Note that if you pick the last option, you don't get extra time in your allotted presentation slot for setting up amps, putting together an instrument, tuning, etc., so if you want to perform make sure you have a well-thought out plan of how to avoid wasting time based on that choice.

The Critique

Everyone is expected to attend everyone else's presentations. After seeing all the other presentations, you will pick another group's project to critique on your own (i.e, this part is NOT a group submission). If it was an excellent project, what made it excellent? If the project seemed below the bar, what were its deficiencies? Was the compositional approach "algorithmically interesting" enough to be worthy of a final project?

You will write 1-2 paragraphs critiquing one of the other groups. After the critique of the other group, you will write a paragraph critiquing your own project and your team: were you able to meet your goals? Did everyone share the work equally? Did anyone on the team barely do any work?

Submission and Grading

Submitting

Your final code, `README`, and presentation slides will be submitted as a group in two parts:

- Presentation Slides: upload to Gradescope as a `pdf` file.
- Code: submit on Halligan with command `provide comp150amc final README project.py`

Make sure to select all team members when submitting on Gradescope.

You are required to have a `project.py` file that I will run based on your `README` directions, but you are welcome to provide additional files if `project.py` depends on them (e.g., for modularity). Simply list all the necessary files after `project.py` in the `provide` command above.

You will also submit, on your own, your critique to Gradescope as a `pdf` file. This will be a separate "assignment" on Gradescope.

Grading

Your final presentation will be graded out of 10 points based on the following rubric:

Points	Description
2	The presentation didn't go over time.
3	The project's algorithmic approaches are clearly explained.
3	Visualizations accurately and clearly show where the algorithms are "hiding" in the generated music.
2	The final code is run, live, to produce the presented composition.

Your final project itself will be graded holistically, using the following rough scale based on the code, `README`, and content captured in the presentation:

- A: Outstanding work. An "A" group completed significant work to execute the assignment and showed an exceptional level of insight in both the implementation and explanation of their project. The presentation and code documentation make a strong case for the algorithmic interest of the compositional approach.

- B: Solid work. A “B” group’s work and explanation are strong and there is some algorithmic interest in the compositional approach.
- C: Weak work. A “C” group completed the bare minimum amount of work to call their submission a “final project.” Little algorithmic interest in the compositional approach, or just reusing approaches that were already covered in class.

Your critique will also be graded holistically based on whether it meets the requirements stated earlier in the spec and seems to have thought put into it.