Outline

1. My Objectives
2. My Approach
3. The Results
Motivation

- I’ve loved classical music since childhood.
- In junior high I was making up classical music in my head, but I became a flute player, not a composer.
- I was afraid I had no talent, and wouldn’t be able to master music theory.
- In 1978 Martin Gardner got me interested in the beauty of fractals.
- I saw how a snowflake curve could be interpreted as a piano roll.
- I could compose without having to get into that competitive, technical stuff.
- In the end, fractals just dragged me right back into that competitive, technical stuff . . .
Irreducibility

- In art music it’s pointless to automate *anything* a composer *might* do better by hand.
- Luckily, many algorithms generate outputs that can’t be predicted by reading the algorithm.
- The technical term is “computational irreducibility.”
- Irreducibility generates the beauty of Nature: branchings, flows, clouds.
- Using irreducibility to generate abstract art illuminates Aristotle’s proposition that art is the imitation of Nature.
- Sometimes, just mapping an irreducible algorithm onto pitch $\times$ time makes a great piece . . . usually, *not*.
- I decided the spaces score generators operate in should *already* have musical structure.
Mathematical Music Theory

- In spite of serial music’s tiny audience, contemporary music theory is heavily influenced by “atonal theory.”
- Atonal theory is the second time in history a major art form was given a mathematical basis. (The first was perspective in painting.)
- In the late 20th and early 21st century, American and European theorists have mathematized not only atonal theory, but also the theory of voice-leading (“neo-Riemannian theory”) and functional harmony.
- Once theory is mathematical, it can be used to write software that generates scores.
I compose music by writing computer programs.
The program runs once, and writes a soundfile.
I seldom assemble, and never post-edit.
I make way too many bad pieces, but I have made some I like.
Sometimes I make a piece that would be good, if only it didn’t have a few “clunkers.”
I’m trying to use mathematical music theory to make it likelier for the score generator to move from one moment to the next in a musically well-formed way.
In music, “well-formed” just means “easy to hear.”
Generating

- I favor score generators that are “universal,” so they can produce any set in their space.
- I favor score generators that depend on numerical parameters, so they can be mapped.
- I’m now basing generators in spaces structured by mathematical music theory.
- I use Fiore and Satyendra’s Generalized Contextual Group to generate progressions from voice-leading transformations.
- I use Tymoczko’s chord space $OP$ factored into $P \times I \times T \times V$ to compute progressions with revoicings.
I write everything in C++. Each piece is one file that builds with a single command.

I embed Csound and a Csound orchestra in my pieces, and aim for a finished level of sound design and mastering.

I have automated absolutely all production steps except writing the score generators and Csound instruments.

In my text editor, I tinker with a few lines of C++, press a key, and get a mastered piece.

Yes, this approach would work fine with RTcmix.
Some Pieces

- **Cloud Strata**, 1998 (actually track 12 not 1). Lindenmayer system mapped onto pitch \( \times \) time.
- **Triptych**, 1998 (actually track 11 not 2). Chaotic dynamical systems mapped onto pitch \( \times \) time, filtered by pre-determined chords.
- **Two Dualities**, 2010. Lindenmayer system mapped onto the Generalized Contextual Group.
- **Blue Leaves 4e**, 2012. Recurrent iterated function system mapped onto pitch \( \times \) time, filtered by selected chords.
- **Untouching 1**, 2012. Recurrent iterated function system system computing a fractal interpolation function from \( t \) onto \( P \times I \times T \times V \).

