C-major

A Music Production Language
Musical Composition language with C-Style syntax
Common constructs - loops, conditionals
Dedicated types to represent musical elements
Designed to abstract computerized composition in a context of Turing-complete features
Pitches
Tutorial

1. Compile source code: make
2. Run test suite: ./test.sh
3. Write your program: *.cmaj
4. Compile into CSV “bytecode” using the cmajor compiler
5. Use Java CSVPlayer to render audio output
Program (*.cmaj) → Compiler (cmajor) → Bytecode (*.csv) → CSVPlayer
Language Implementation

- `<cmaj file>`
- Scanner: scanner.mll
- Parser: parser.mly
- Semantic Analyzer / Translator: compile.ml
- MIDI Converter: CSVPlayer.java, NotesPlayer.java
- `<MIDI file>`
Data Types

- Int
- Dur
- Pitch
- Note
- Chord
- Phrase
- Score
Operators and Type Inferences

- **Arithmetic**: Addition, Subtraction, Multiplication, Division
- No floating point numbers: all fractions treated as rational numbers consisting of two integers
- **Advanced array manipulation**: Repeat/Fill, Concatenate
- **Layer**: Two musical types to be rendered simultaneously in audio output
- **Concatenate**: Provides audio sequencing
Testing

test.sh

- System testing framework using shell script
- Compares each .cmaj program in the Tests directory with a reference file of its expected .out output
- Ends report by displaying number of passed tests and total tests run
- Collects failed and total test outputs
- Tests: interpreter, parser, semantics
  - [arithmetic, equality, array, pitch, array, array concat, function, compose, play, layer, blocks, statements, comments, etc]
Demos

Row, Row, Row Your Boat

Demonstrates loops, assorted operators

An old favorite

Shepard Scale

Demonstrates functions, loops, conditionals

A “sonic barber pole”
Summary and Lessons Learned:

- Weekly Meetings and Waffle
- Share and Listen to Ideas
- Ocaml: Do a lot with a little