A MINIMAL FRACTAL MUSIC
FRACTAL MUSIC COMPOSITION LANGUAGE

Jake Amonette, Tal Giat, Brian Hsieh, Eugene Yao
Team Mondegreen
Eugene
Language
And
Tools
Guru
Jake Amonette

Systems

Integrator
PM

Architect

Tal →
MOTIVATION
DIFFERENT
FRACTAL
CONTEXT FREE ART
STARTSHAPE START

RULE START{ SS1{ $ 1.5 Y -.13} }

RULE SS1 {
    BASE{$ 1.3}
    BARNEY {}
    FRED {}
}

RULE BASE {
    SQUARE {B 0}
    SQUARE {S 0.9925 B 1}
}

RULE FRED {
    SQUARE {B 0}
    SQUARE {S 0.9925 B 1}
    FRED {R -3 S 0.98}
}

RULE BARNEY {
    SQUARE {B 0}
    SQUARE {S 0.9925 B 1}
    BARNEY {R 1 S 0.98}
}
!PROGRAMMER
DETERMINISTIC
STEVE REICH <AMFM< MOZART
Language Description

- AMFM
  - Object based music fractal composition language
    - Seed-based
    - Self-similar manipulations
  - Outputs music file
Sample Program

seed pianotest
{
  PIANO; // 6 bars of flute, 1 bar per line
  A4,1/4; C5,1/4; E5,1/8; G5,1/8; F4,1/4;
  E5,1/4; C5,1/4; A4,1/2;
  A4,1/4; C5,1/4; E5,1/8; G5,1/8; F4,1/4;
  E5,1/4; C5,1/4; A4,1/2;
  C5,1/4; D5,1/4; G5,1/8; A5,1/8; Ab4,1/4;
  G5,1/4; D5,1/4; C5,1/2;
}

seed flutetest
{
  FLUTE; // 1 bar of flute, 1 bar per line
  A4,1/8; C5,1/8; E5,1/16; G5,1/16; F4,1/8; E5,1/8; C5,1/8; A4,1/4;
}

start
{
  play(pianotest, 1, 6);
  SSRhythm(flutetest, 3, 4);
}
Sample Program

seed pianotest
{
    PIANO;  //6 bars of piano, 1 bar per line
    A4,1/4; C5,1/4; E5,1/8; G5,1/8; F4,1/4;
    E5,1/4; C5,1/4; A4,1/2;
    A4,1/4; C5,1/4; E5,1/8; G5,1/8; F4,1/4;
    E5,1/4; C5,1/4; A4,1/2;
    C5,1/4; D5,1/4; G5,1/8; A5,1/8; Ab4,1/4;
    G5,1/4; D5,1/4; C5,1/2;
}
seed flutetest
{
    FLUTE; // 1 bars of flute, 1 bar per line
    A4,1/8; C5,1/8; E5,1/16; G5,1/16; F4,1/8; E5,1/8; C5,1/8; A4,1/4;
}
Sample Program

```
start
{
    play(pianotest, 1, 6);
    SSrhythm(flutest, 3, 4);
}
```
seed pianotest
{
  PIANO; //6 bars of flute, 1 bar per line
  A4,1/4; C5,1/4; E5,1/8; G5,1/8; F4,1/4;
  E5,1/4; C5,1/4; A4,1/2;
  A4,1/4; C5,1/4; E5,1/8; G5,1/8; F4,1/4;
  E5,1/4; C5,1/4; A4,1/2;
  C5,1/4; D5,1/4; G5,1/8; A5,1/8; Ab4,1/4;
  G5,1/4; D5,1/4; C5,1/2;
}

seed flutetest
{
  FLUTE; //1 bar of flute, 1 bar per line
  A4,1/8; C5,1/8; E5,1/16; G5,1/16; F4,1/8; E5,1/8; C5,1/8; A4,1/4;
}

start
{
  play(pianotest, 1, 6);
  SSRhythm(flutetest, 3, 4);
}
Translator Architecture

AmFm Source → Lexer → Tokens → Parser → Antlr → AST

Lexer: AmFmLexer.java
Parser: AmFmParser.java

Antlr

Java Class → Code Generator → AST → Code Optimizer

Code Generator: AmFmGenerator.java
Code Optimizer: AmFmOptimizer.java

String Template

Java Sound API

Fractal Composer
Systems Integration

- Music generated by **Fractal Composer** by Myron Marston
- Open source fractal music engine
- www.fractalcomposer.com
Fractal Composer

- Fractal music Java library
Fractal Composer

- AMFM generates a Java class that uses Fractal Composer
- Fractal Composer generates musical output
Fractal Composer

- Simple, right?
Problems!
Fractal Composer

- Normally accessed via web front end

- Unsuitable for our use. Also...
Fractal Composer

- UI is cumbersome for creating large/complex compositions
Fractal Composer

- Can’t read a UI like you can a program (or musical score)
Fractal Composer

□ Limitations!
Fractal Composer

- Cannot have more than one seed per composition
Fractal Composer

- No notion of sequence
- AMFM allows composer to start or stop playing seed arbitrarily
Fractal Composer

Solution!
Fractal Composer

- Extensive modifications to library
Fractal Composer

- Refactored structure of library
Fractal Composer

- Added multi-seed capability
Fractal Composer

- Added basic sequencing
Fractal Composer

- Modified API to allow generic front end
# Environment and Support Tools

<table>
<thead>
<tr>
<th>IDE</th>
<th>Testing</th>
<th>Idea Platform</th>
<th>Translator Framework</th>
<th>Lang</th>
<th>Documentation + Source Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eclipse + AntlrIDE</td>
<td>gUnit + JUnit</td>
<td>Chalkboard</td>
<td>Antlr + StringTemplate</td>
<td>Java 1.6</td>
<td>Google Code: Wiki, SVN</td>
</tr>
</tbody>
</table>

- Eclipse + AntlrIDE
- gUnit + JUnit
- Chalkboard
- Antlr + StringTemplate
- Java 1.6
- Google Code: Wiki, SVN
Test Plan

Code check in/out
Test Plan

- Write new code
- Code check in/out

Write new code
Code check in/out
Test Plan

1. Code check in/out
2. Write new code
3. Automated regression testing
Test Plan

- Code check in/out
- Write new code
- Automated regression testing
- Black box testing
Test Plan

1. Code check in/out
2. Write new code
3. Black box testing
4. Automated regression testing
5. All tests passed
gUnit

- gUnit, a unit testing framework forANTLR
- SampleTest.gunit

```java
@header{package org.mondegreen.amfm.compiler;}

//test lexer

COMMENT:
"//hello\n" OK
"// hello world\n" OK
"/** too complicated */" FAIL
```
gUnit

C:\Users\Brian Hsieh\Documents\COMS4115\amfm>java -cp lib\antlrworks-1.2.3-amfm.jar;bin org.antlr.gunit.Interp test\org\mondegreen\amfm\compiler\SampleTest.gunit
executing test suite for grammar:AmFm with 3 tests
0 failures found:
Tests run: 3, Failures: 0
HARD
Constructs a midi track based on the given note list.

```java
protected void constructMidiTrack(NoteList noteList) {
    MidiNote thisMidiNote, lastMidiNote = null;
    Fraction startTime = new Fraction(0, 1);
    // get a default instrument if we're not passed one...
    // make each track be on a different channel, but make sure we don't go over our total number of channels...
    int numTracks = sequence.getTracks().length;
    // The 1st track should be the tempo/key sig/time sig track
    int midiChannel = numTracks;
    Number numTracks = 0; numTracks = 1;
    noteList.getInstrument() = MidiNote.MAX_CHANNEL;
    Part part = new Part(this.pieceNotation, instrument);
    for (Note thisNote : noteList.getNotesWithNormalizedRests()) {
        // update our part section if necessary...
        if (partSection == null) {
            partSection = new PartSection(part, thisNote); parts.add(partSection); sources.add();
        } else if (lastNote != null) {
            lastNote = new PartSection(part, lastNote); lastNote = new PartSection(part, lastNote);
        }
        int timeSignature = partSection.getTimeSignature();
        // in Midi, the tick resolution is based on quarter notes, but we use whole notes...
        int midiNotesPerWholeNote = convertMidiTickUnitFromQuarterNotesToWholeNotesInt(sequence.getResolution());
        MidiNote thisMidiNote = thisNote.convertToMidiNote(startTime, midiNotesPerWholeNote, midiChannel, true);
        if (lastMidiNote != null) {
            noteList.getInstrument() = MidiNote.MAX_CHANNEL;
            if (!thisMidiNote.getPitch().equals(lastMidiNote.getPitch()) || lastMidiNote.getNormalizedNote().getScaleStep() != thisMidiNote.getNormalizedNote().getScaleStep()) {
                lastMidiNote = new MidiNote(startTime, midiNotesPerWholeNote, midiChannel, true);
                lastMidiNote = thisNote.convertToMidiNote(startTime, midiNotesPerWholeNote, midiChannel, true);
            }
        }
    }
}
```
myronmarston / fractal_composer

Description: The algorithm used by fractalcomposer.com

Welcome to the fractal_composer wiki!

Last edited by myronmarston, 18 days ago

Hosting provided by our partners at Engine Yard