A harmonic oscillator simulation using Groovy

Angela Popa, Nga Pham

DSL Engineering, SS2018
Angela.Popa@student.uibk.ac.at
Nga.Pham@student.uibk.ac.at

Motivation

- Main Purpose
  - Create DSL for Simulating Harmonic Oscillator
- Main functionalities
  - Harmonic Simulator
    - Compose all oscillator waveforms into a harmonic
  - DSL Editor
    - Enable user to define Components, their initial values and relationship

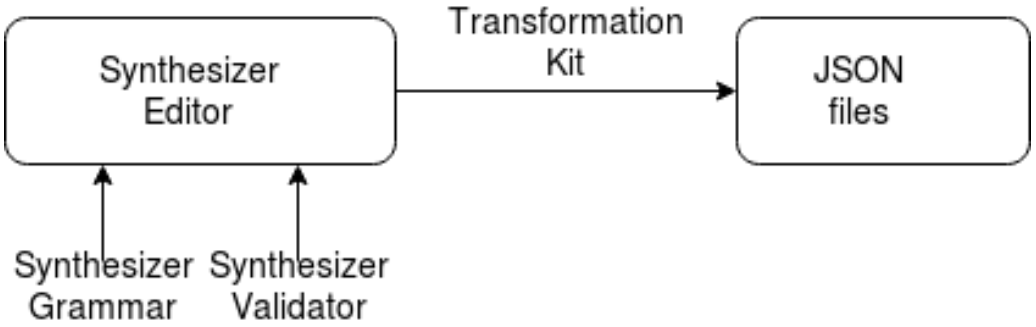
Environment

- Groovy
  - Metaprogramming
    - Register a new method on the fly (ExpandoMetaclass)
  - SwingBuilder
    - Create UI components
- Jsyn API
- JSON
  - Groovy script to read/write JSON data

Why did we choose Groovy instead of other tools?

Application Structure

- Step 1: DSL



- Step 2: Implement the Simulator

User Actions

- Define Components
- Configure Components – the initial state
- Define Relationships between Components
- Define the default Operation for the Sound Composition

User Actions – Define Components

- Define Controllers and Filters

```
controls
    .add(new RotaryKnob(
            type: 'knob',
            name: 'myKnob',
            digits: 5
        ))
```

```
controls
    .add(new Slider(
        type: 'slider',
        name: 'mySlider'
        ))
```

```
filters
    .add(new LinearRamp(
        name: 'frequencyRamp',
        type: 'LinearRamp',
        input: new LinearRampInput(
            minimum: 50.0,
            actualValue: 300.0,
            maximum: 10000.0
        ),
        connectsTo: 'frequency',
        time: new LinearRampTime(
            duration: 0.2
            )
        ))
```

User Actions – Define Components

- Define Oscillators

```
oscillators
    .add(new Oscillator(
            name: 'myFirstOsc',
            type: 'SineOscillator',
            amplitude: new Amplitude(
                minimum : 0.0,
                maximum : 1.0,
                defaultValue : 0.5
            ),
            frequency: new Frequency(
                minimum : 50.0,
                maximum : 10000.0,
                defaultValue : 300.0
            )
    ))
```

User Actions – Add Relationships

- Add Connection

```
connections
     .add(
          new Connection(
               filter: 'amplitudeRamp',
               fromController: 'myKnob',
               toOscillator: 'myFirstOsc'
          )
     )
```

User Actions – Sound Composition

- Define the default Operation for the Sound Composition

```
waveformOperations
    .add(
        new Operation(
            name: 'Division'
        )
    )
```

DEMO

-

Challenges

- Deciding which path to take
  - code generation or
  - wrapper approach (winner!)
- Separate DSL editor from logic
  - Reuniting via binding
  - Went for persisting editor data into json
- Using Java Enum class in both editor groovy scripts and logic groovy scripts
  - Went for using enums as groovy script &
  - loading via groovy class loader

Conclusion & Future Plans

- Benefit for defining a DSL is clear
  - Jsyn API can be challenging
- Ideas for future work
  - Extend the Demo with a build-in song
  - Experiment more with different Jsyn Components (Unit Generators)