COMPUTER MUSIC

Languages and Systems

Prof. Fabio Antonacci
Prof. Marco Olivieri

POLITECNICO DI MILANO
Music and Acoustic Engineering

Homework 2 – Assignment 3

Subtractive Synthesizer with JUCE

Work Team

Lelio Casale
Marco Furio Colombo
Marco Muraro
Matteo Pettenò

OranJam

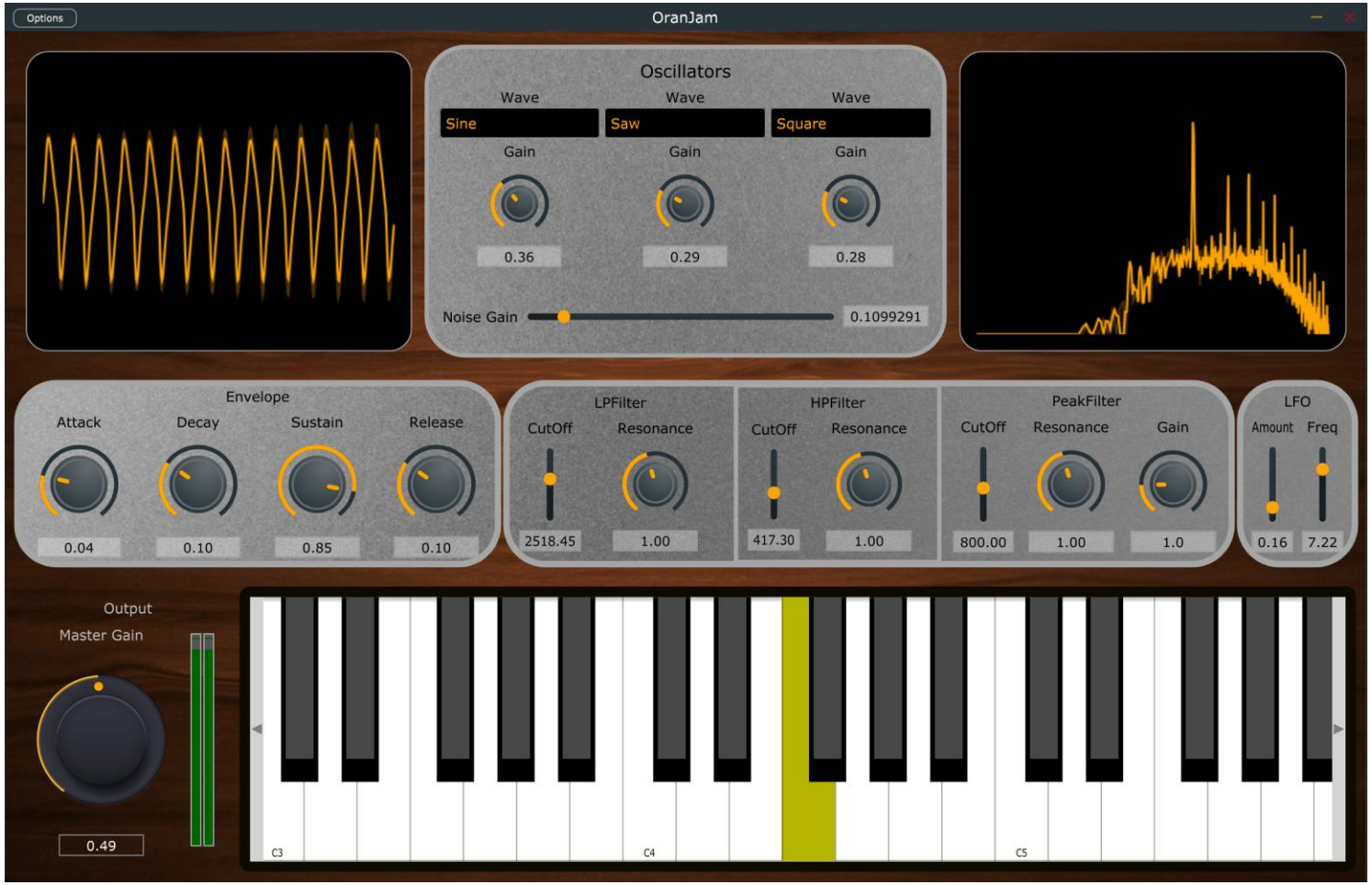
Subtractive Synthesizer

Controls divided in six groups

1. Oscillators
2. Envelope (ADSR)
3. Filters
4. Low Frequency Oscillator
5. Master Volume
6. keyboard

Three Visualization Elements

1. Volume meter
2. Oscilloscope
3. Spectrum analizer



Graphical User Interface

- Retro looking interface that includes functional sound visualization.
- **Foley_gui_magic**: an open source JUCE Library by Daniel Waltz.
- A CSS cascading stylesheet in xml file defines rules for the appearance and behaviour of the GUI.
- ***MagicPluginEditor*** replaces the *PluginEditor*, with data fetched from the *AudioProcessorValueTreeState* and the xml.
- Great scalability and flexibility, with no big drawbacks.
- Poor Documentation lead us to needing to inspect the source code.

Build System

Projucer

Pros

- Quick Setup
- User-friendly GUI

Cons

- Hard to link external code/libraries
- Doesn't scale well

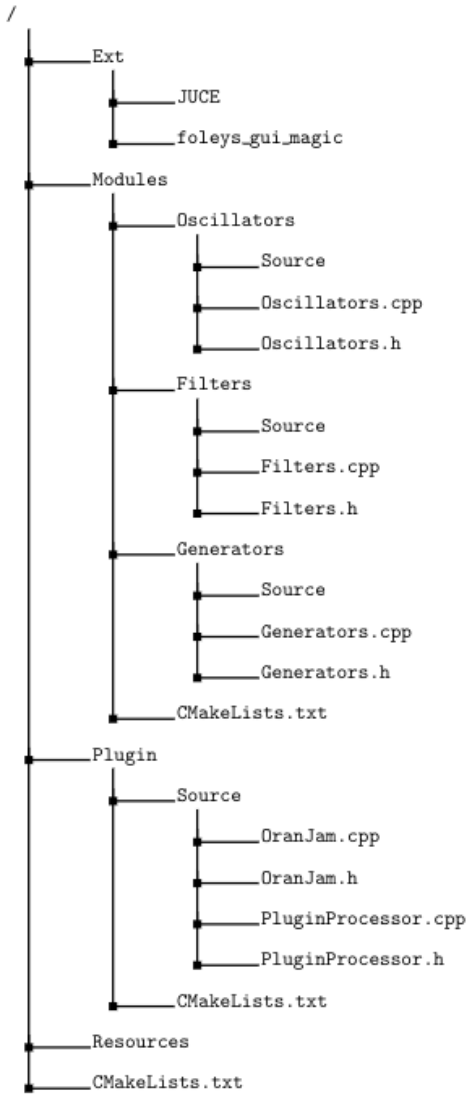
CMake

Pros

- Wider IDE integration
- General code modularisation (not only JUCE modules)
- Simpler CI integration
- Easier integration with other projects that use CMake
- Project distribution with package managers (Vcpkg and Hunter)

Cons

- Scripting language
- Not so easy to deal with at first



SW Architecture

Plugin (Synth core)

- OranJam
- Plugin Processor

Modules

- Oscillators
- Filters
- Generators

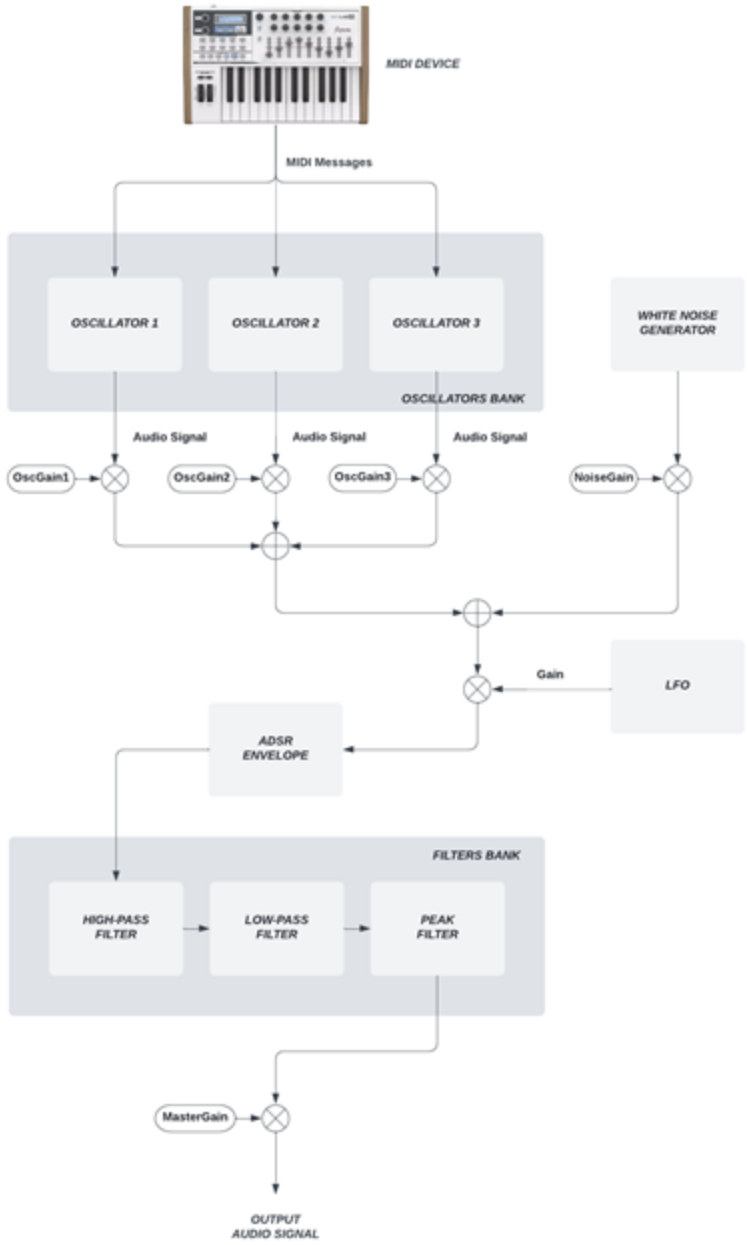
Ext

- JUCE
- Foleys GUI Magic

Resources (Binary Data)

- Images
- XML
- etc…

Signal Flow


MIDI DEVICE
MIDI Messages
OSCILLATOR 1
OSCILLATOR 2
OSCILLATOR 3
OSCILLATORS BANK
WHITE NOISE GENERATOR
Audio Signal
Audio Signal
Audio Signal
OscGain1
OscGain2
OscGain3
NoiseGain
Gain
LFO
ADSR ENVELOPE
FILTERS BANK
HIGH-PASS FILTER
LOW-PASS FILTER
PEAK FILTER
MasterGain
OUTPUT AUDIO SIGNAL

OranJam Implementation

For **OranJam** synth implementation different JUCE classes were exploited.

JUCE::Synthesiser

A Synthesiser object models a monophonic or polyphonic synth. A Synthesiser needs a SynthesiserSound and a SynthesiserVoice.

JUCE::SynthesiserSound

Basically, describes one of the sounds that can be played by a Synthesiser object. This is just a passive class that defines the sound. The actual audio rendering for a sound is accomplished by a SynthesiserVoice.

JUCE::SynthesiserVoice

Models the single voice of a Synthesiser. A voice can play a sound at a time. A Synthesiser holds an array of one or more voices.

OranJam Implementation

OranJam synth were implemented within OranJam class, which inherits from *JUCE::Synthesiser*.

Two additional classes modeling a sound and the single voice were implemented as subclasses of OranJam class.

class OranJam : public JUCE::Synthesiser

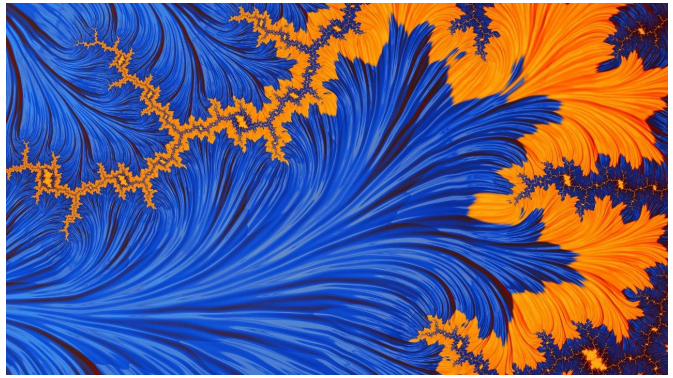
Implements OranJam synth.

class SynthSound : public JUCE::SynthesiserSound

Holds the *AudioProcessorValueTreeState* representing the synth state.

class SynthVoice : public JUCE::SynthesiserVoice

Accomplishes the audio rendering of the single voice.



Audio Processing External Modules

Three main blocks

Oscillators

Three different types of waveforms: Saw, Square and Sinusoidal

Generators

Contains an ADSR envelope, an LFO and the white noise

Filters

IIR high-pass, low-pass and peak-filters

Implemented with the methods makeHighPass, makeLowPass and makePeakFilter of the *dsp::IIR::Coefficients* JUCE class

DEMO VIDEO

OranJam.mp4

Thank You!

Lelio Casale
Marco Furio Colombo
Marco Muraro
Matteo Pettenò

OranJam

Audio
Synthesis
Enterprises