POLITECNICO MILANO 1863

Computer Music - Languages and Systems

Project:

LadyfingerS Morph Delay

Beretta Pietro – Di Nucci Alessandro – Forlani Serena – Karam Anthony – Messina Giorgio – Neble Manuel – Parlato Domenico

Outline

- **System Overview:** The hybrid interaction pipeline.
- **SuperCollider:** Granular engine & OSC routing hub.
- **JUCE Audio Engine:** DSP chain & Macro-Morphing paradigm.
- **Tiramisù Look & Feel:** Custom UI design & visual feedback.
- **Processing:** Generative visuals via "Musical Game of Life".
- **Live Performance:** Real-time system demonstration.

SuperCollider – audio engine & OSC routing hub

- **Modular Architecture**
  Strict separation between MIDI input, explicit musical state logic (note_state.scd), mapping, and sound synthesis.

- **Smart State Management**
  Real-time computation of normalized velocity, polyphony (max 6 voices), legato/repeated note recognition, and register classification (low/mid/high).

- **Multi-layer Synthesis**
  8 independent sound generators (Sine, Saw, FM, Glitch, etc.) dynamically triggered via Touchpad.

- **OSC Communication Hub**
  Parallel routing of high-level processed data to JUCE (port 9001) and Processing (port 12000) for perfect audio-visual sync.

JUCE – advanced audio processing & modulation

- **Dynamic Mophing Engine**
  A macro control («Espresso») that simultaneously interpolates 5 delay parameters (time, feedback, damping, wet, dry) for seamless sonic transitions. [DelayModulationEngine]

- **Organic Modulation**
  Multi-waveform LFOs and real-time audio amplitude tracking to auto-modulate feedback and damping without manual intervention. [LFOMatrix & EnvelopeFollower]

- **Granular Pitch Shifter**
  Dual read-head algorithm with offset and sinusoidal crossfade, designed to prevent acoustic artifacts during frequency shifting.

- **Custom UI**
  User interface drwan entirely via C++ code (Graphics API), featuring an integrated Spectrum Analyzer and a reactive Envelope Meter.

JUCE – OSC integration & custom UI

- **OSC Communication**
  Real-time listening on port 9001 to map /cc (control change) and /pb (pitch bend) messages from SuperCollider directly to plugin parameters.

- **Dynamic Microtonality**
  Pitch bend (MPK joystick) mapped to a -100/+100 cents range (pitch fine) and Knob 8 mapped to a -24/+24 semitones range, sent to the granular processor.

- **Tiramisù look&feel**
  Custom vector GUI written from scratch (pure juce::Graphics), with custom sliders.

- **FFT Spectrum Analyzer**
  Real-time frequency spectrum visualizer (low/mid/high) using juce_dsp, featuring a logarithmic scale and decay mapping for smooth, natural visual feedback.

Processing: musical «game of life»

- **Emergent Visuals**
  A musical reinterpretation of Conway's Game of Life. The simulation doesn't start randomly; MIDI notes act as seeds, spawning new living cells colored by their pitch.

- **Rule-bending DSP**
  Plugin effects directly alter the biological rules of the environment.

- **Time & chaos**
  Delay controls the simulation's evolution speed (generation updates), while Drive introduces probabilistic chaos into cell survival and death.

- **Fertility and Hostility**
  Feedback boosts cell fertility (allowing births with 2 or 4 neighbors instead of just 3), and Damping makes the environment hostile, increasing the death probability of stable cells.

System architecture & interaction novelties

- **Hybrid Pipeline**
  AKAI MPK → SuperCollider (OSC/Granular) → VB-Audio Cable → JUCE (FX) & Processing (Visuals)

- **Macro-Morphing Paradigm**
  A single control simultaneously interpolates 5 delay parameters (time, feedback, damping, wet, dry), reducing cognitive load.

- **Direct Microtonality**
  Horizontal joystick (pitch bend) bypasses SC processing and is mapped directly in JUCE to a +-100 cents range, allowing continuous detuning.

- **Rule-Bending Visualization**
  Mapping DSP effects to the fundamental rules of a cellular automaton (Game of Life), creating a deeply integrated audio-visual ecosystem.


FILE EDIT ADD PATTERNS VIEW OPTIONS TOOLS HELP
Fruity Wrapp_EchoMorph
Pattern 1
EchoMorph (LadyFingerS)
Presets
Ladyfingers
Morph Delay
SuperCollider OSC - Tiramisu Edition
ESPRESSO
Espresso
PANNA
0.709
Wet
Dry
Time ms
Feedback
0.394
0.331
122
0.000
Damping
Drive
Pitch
3377
8.48
0.543
Ping Pong
Morph ON
Drive ON
Output Spectrum
Mixer - LadyFingerS
EchoMorph
Slot 2
Slot 3
Slot 4
Slot 5
Slot 6
Slot 7
Slot 8
Slot 9
Slot 10
Equalizer

Future envelopments: DAW Integration

Full VST/AU plugin integration

- Compiling the JUCE engine as a standard native plugin format (VST3, AU) for direct DAW hosting.
- Routing multi-channel audio streams and parallel OSC data directly inside the DAW environment.
- Challenge: Managing precise clock synchronization and minimizing latency between SuperCollider and the host DAW.

Future envelopments: DAW Integration

Hypothesis B: Resampling & Audio Export Workflow

- Implementing a dedicated real-time recording module within the standalone application.

- Exporting high-quality audio files (stems in .wav format) captured from the live morphing performance.
- Seamlessly importing files into any commercial DAW for traditional arrangement, layering, and mixing.
- *Advantage:* Immediate availability of highly dynamic, organic textures and layers for modern music production.

Declares the asynchronous recording infrastructure within JUCE.

PluginProcessor.h

```
std::unique_ptr<juce::AudioFormatWriter::ThreadedWriter> threadedWriter;
juce::TimeSliceThread backgroundThread { "AudioRecordingThread" };
std::unique_ptr<juce::AudioFormatWriter> fileWriter;
bool isRecording = false;
```

instantiates a background thread and a threaded writer object. This setup ensures that writing large audio data to the hard drive is handled on a secondary thread, preventing CPU spikes and protecting the time-critical audio thread from dropouts or glitches.

PluginProcessor.cpp

Positioned at the very end of the processBlock function, it checks if the recording state is active. If true, it dynamically copies the exact audio samples produced by your morphing delay engine and pushes them into the threaded writer's buffer queue before the audio reaches your speakers.

```
void YourAudioProcessor::processBlock (juce::AudioBuffer<float>& buffer,
juce::MidiBuffer& midiMessages) { // ... Il vostro dsp chain esistente (Delay, LFO, Pitch
Shifter) ... // ... L'audio processato ora si trova nel 'buffer' ... if (isRecording &&
threadedWriter != nullptr) { // Scrive il buffer audio nel file in un thread separato per
non bloccare la CPU audiothreadedWriter->write (buffer.getArrayOfReadPointers(),
buffer.getNumSamples()); } }
```

Intercepts the final processed audio stream in real time.