

# Euterpe: A Web Framework for Interactive Music Systems

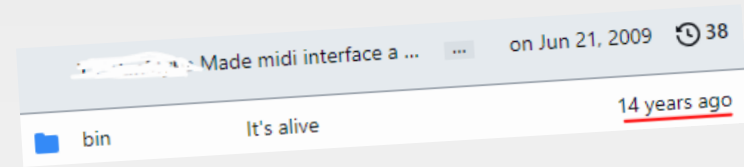
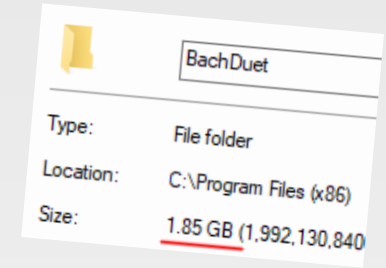
Yongyi Zhang\*, Christodoulos Benetatos\*, Zhiyao Duan

JAES 2023



# Problem Statement

- Research **stops** at **open sourcing** the core algorithms
- Prototype systems that are **not easily accessible**
  - Large executable files
  - Unmaintained codebases
  - Platform dependent implementations
  - Complicated installation processes



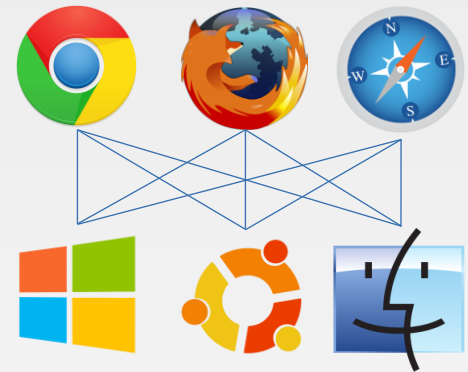
## Installation

OS Support

We currently only support Windows 64-bit.

# A Solution ...

- Promote the development of **web** musical systems
  - Pros
    - Utilize the web's natural cross-platform compatibility
    - End-users are familiar with the browser environment
    - No installation required



# A Solution ...

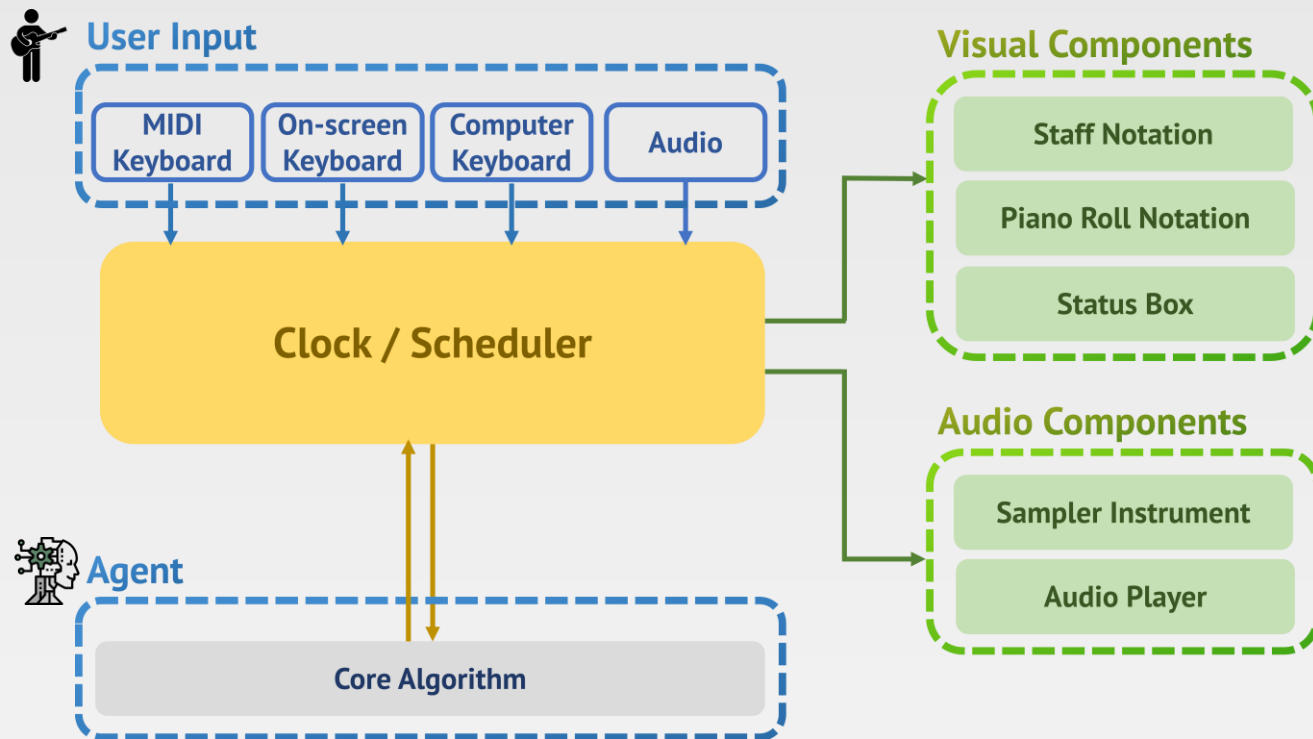
- Promote the development of **web** musical systems
  - Cons
    - Knowledge of web programming is required (JavaScript, CSS, HTML)



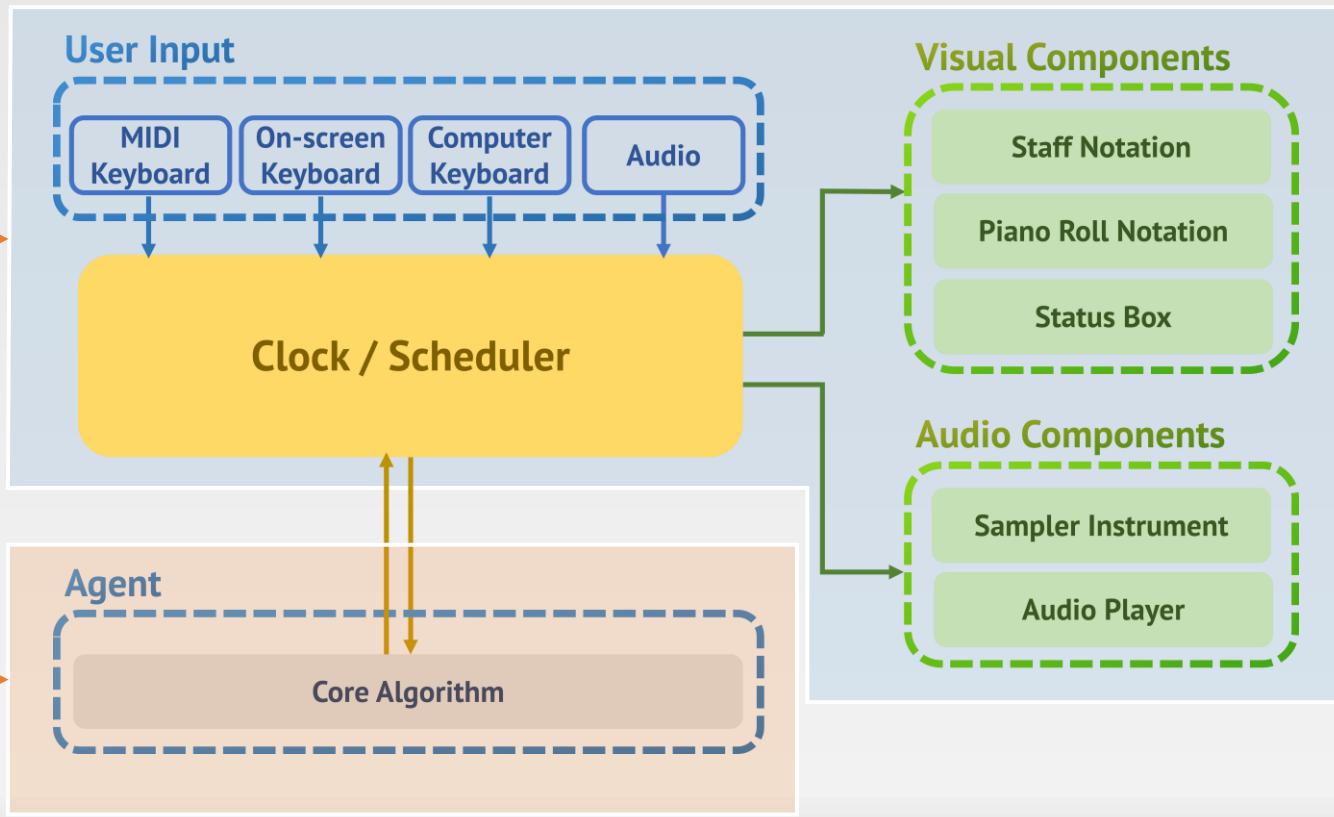
# Euterpe's Goal

- **Alleviate challenges** associated with **web programming**
  - Offer ready-made submodules for common system components
  - Developers focus solely on their system's unique features

# Generic IMS Architecture



# Generic IMS Architecture



# Design

- Modular
  - **Separate** the **Agent code** from the peripheral components
- Configuration files
  - Allow app setup and **customization without** writing **JavaScript**

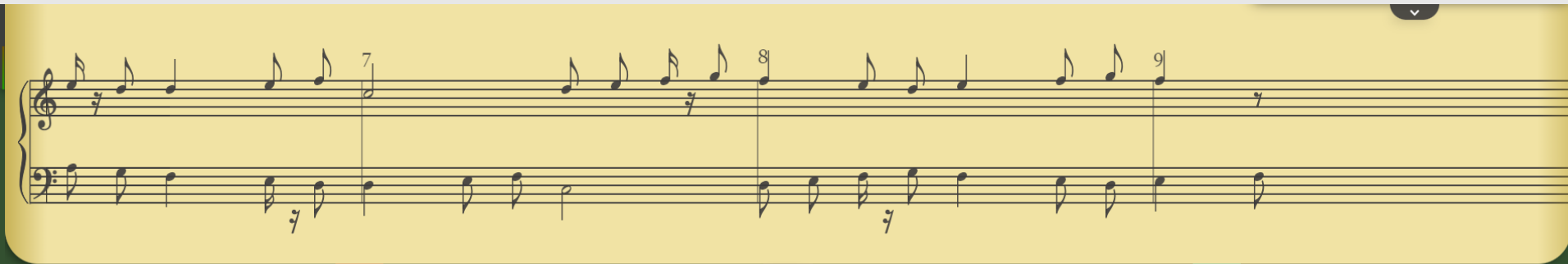


# Visual Components

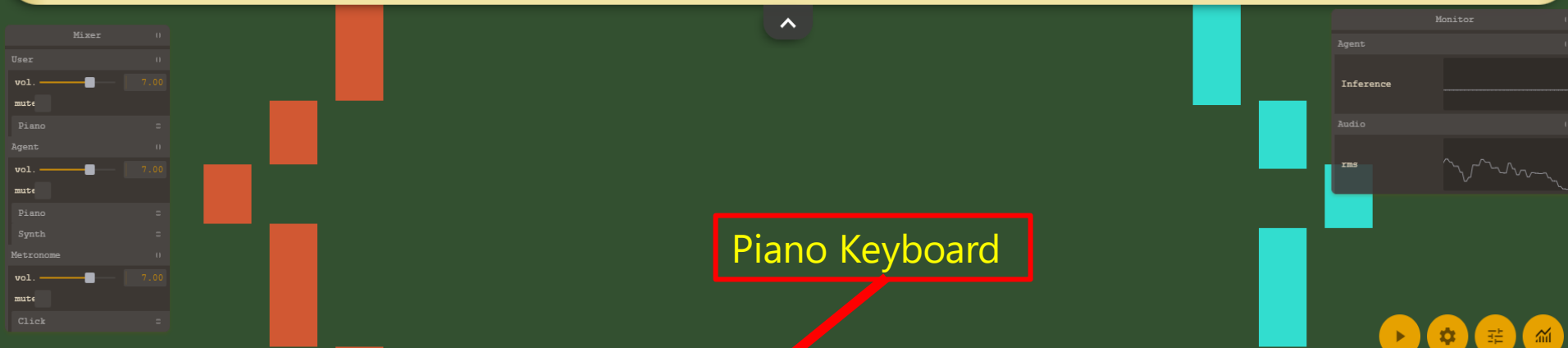
The image displays a music software interface with several key components:

- Score Widget:** A red-bordered box highlights a musical score on a yellow background, featuring a treble and bass staff with notes and rests. A red arrow points from this box to a label.
- Mixer:** A vertical panel on the left side contains controls for various tracks: Mixer, User, Piano, Agent, and Metronome. Each track has a volume slider set to 7.00 and a mute button.
- Monitor:** A panel on the right side shows a waveform graph and labels for Monitor, Agent, Inference, and Audio.
- Keyboard:** A digital piano keyboard is visible at the bottom, with keys labeled with letters (A-Z) and numbers (1-7) and their corresponding musical notes (e.g., A#2, C#3, D#3, F#3, G#3, A#3, S C#4, D D#4, G F#4, H G#4, J A#4, 2 C#5, 3 D#5, 5 F#5, 6 G#5, 7 A#5).
- Score Widget Label:** A red-bordered box with the text "Score Widget" is positioned below the score, with a red arrow pointing to the score area.

# Visual Components



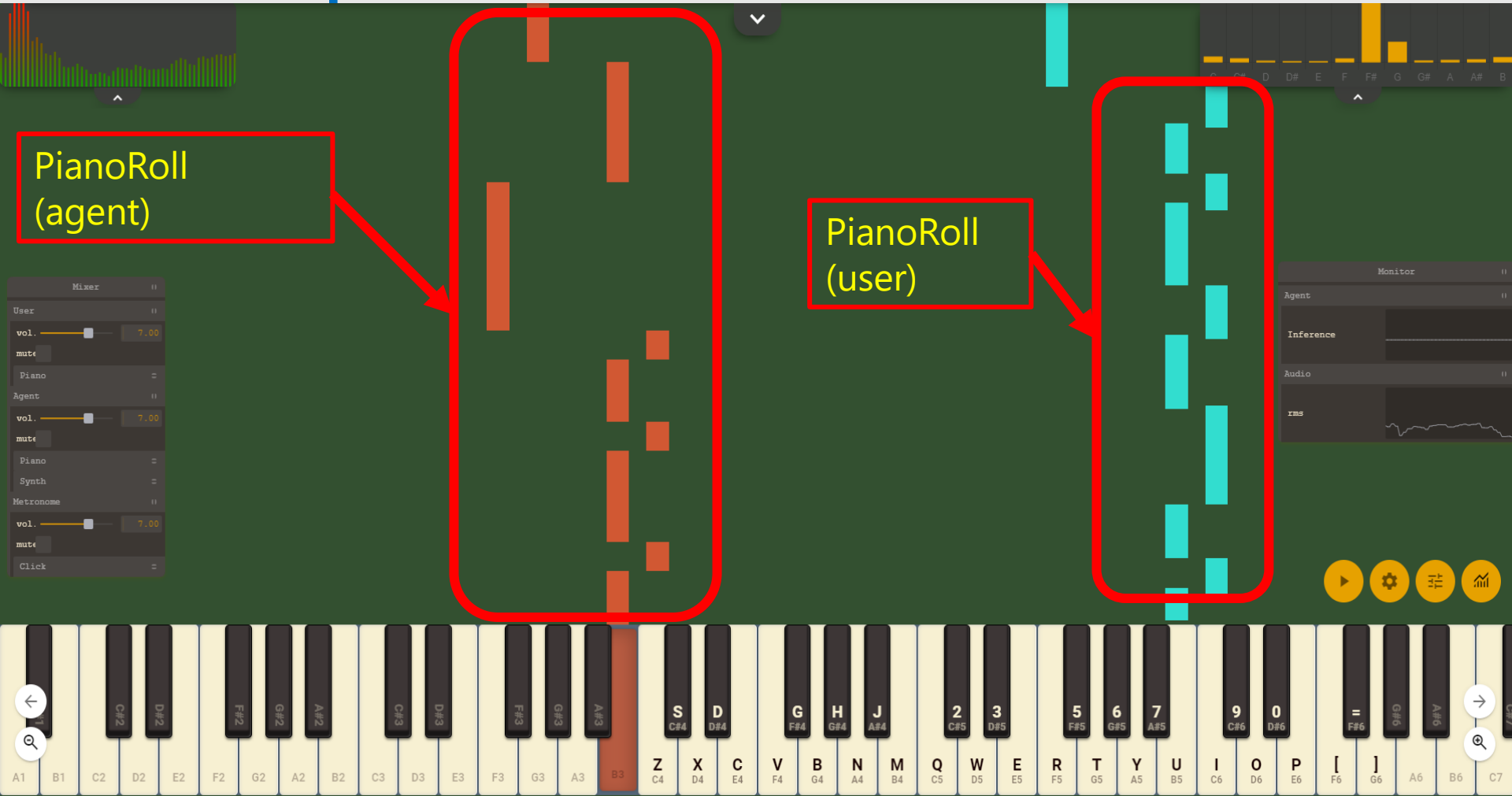
A musical score for piano, showing two staves (treble and bass clef) with notes and rests. The score is displayed on a yellow background. The treble clef staff contains a melody of eighth and quarter notes, while the bass clef staff provides a harmonic accompaniment with chords and single notes. Measure numbers 7, 8, and 9 are visible at the top of the staves.



A screenshot of a Digital Audio Workstation (DAW) interface. The background is dark green with a piano keyboard layout overlaid. The keyboard keys are labeled with letters (A-Z) and numbers (1-7) and their corresponding musical notes (e.g., C#4, D#4, F#4, G#4, A#4, S, D, G, H, J, 2, 3, 5, 6, 7). A red box highlights the piano keyboard area, and a red arrow points to it from a text box labeled "Piano Keyboard". The interface includes a mixer panel on the left with sliders for volume and mute for various tracks (User, Piano, Agent, Piano, Synth, Metronome, Click). On the right, there is a monitor panel with an inference graph and a waveform. The top right corner has a dropdown menu, and the bottom right corner has several control buttons (play, settings, zoom, etc.).

Piano Keyboard

# Visual Components



PianoRoll  
(agent)

PianoRoll  
(user)

# Configuration

```
gui:  
  score:  
    status: true  
  pianoRoll:  
    status: true  
    human: true  
    agent: true  
  keyboard:  
    status: true  
    octaveStart: 2  
    octaveEnd: 6
```

# Visual Components

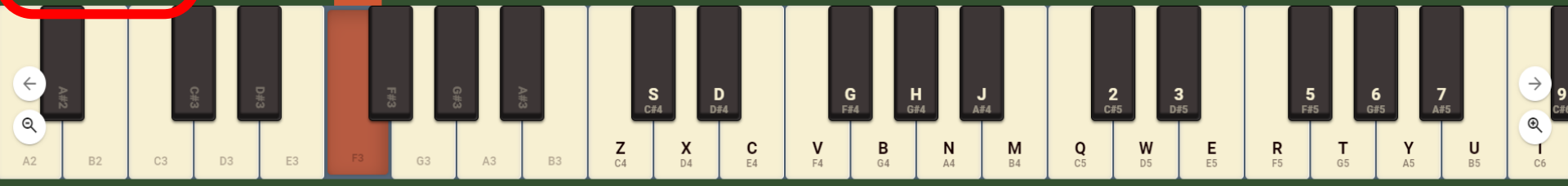


An audio mixer interface with a dark background. It lists several tracks: Mixer, User, Piano, Agent, Piano, Synth, Metronome, and Click. Each track has a volume slider set to 7.00 and a mute button. The interface is highlighted with a red border.

Audio Mixer

Variable Monitor

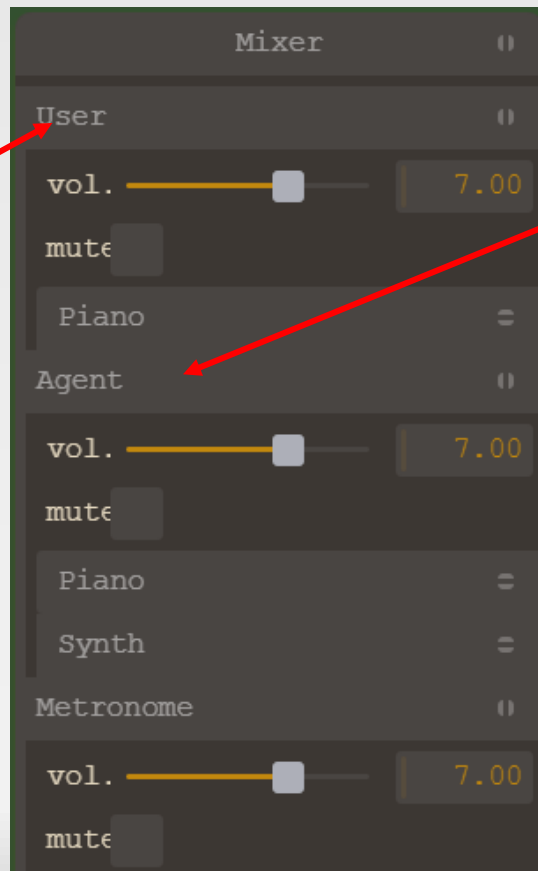
A variable monitor interface with a dark background. It shows sections for Agent, Inference, and Audio. The Audio section contains a waveform graph. The interface is highlighted with a red border.



# Configuration

```

players:
  human:
    label: 'User'
    mute: false
    volume: 5
    instruments:
      - id: "piano"
        label: "Piano"
        mute: false
        volume: 5
        default: true
  
```



```

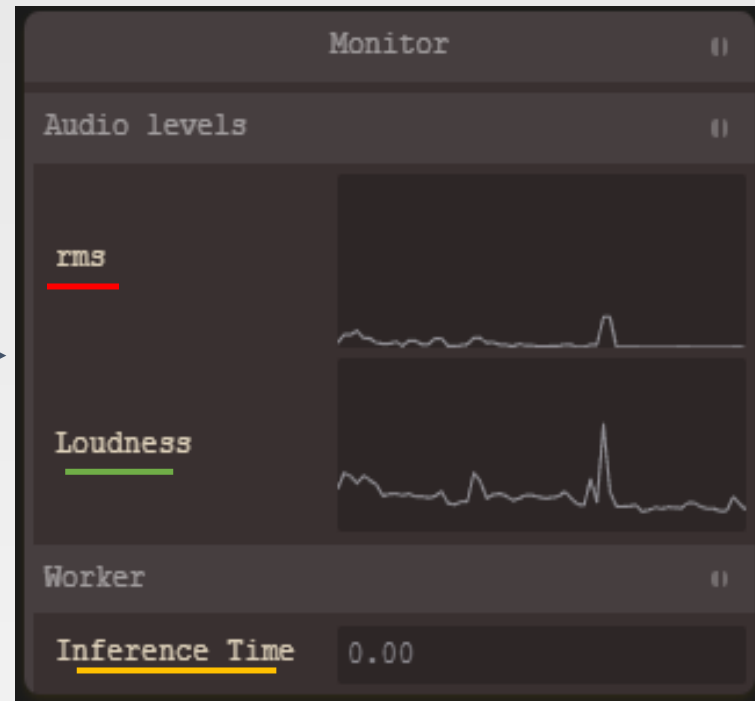
players:
  agent:
    label: 'Agent'
    mute: false
    volume: 5
    instruments:
      - id: "piano"
        label: "Piano"
        mute: false
        volume: 7
        default: true
      - id: "synth"
        label: "Synth"
        mute: false
        volume: 8
  
```

# Configuration

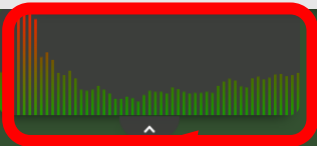
```

1 monitor:
2   title: "Monitor"
3   structure:
4     - label: "Audio levels"
5       parameters:
6         - id: 0 # id's must be unique
7           label: "rms" # choose any name
8           interval: 50 # in ms
9           graph: true
10          min: 0
11          max: 0.2
12         - id: 1
13           label: "Loudness"
14           interval: 50 # in ms
15           graph: true
16           min: 0
17           max: 100
18     - label: "Worker"
19       parameters:
20         - id: 2
21           label: "Inference Time"
22           interval: 100 # in ms
23           graph: false
24           min: 0
25           max: 30

```



# Visual Components



Audio Spectrum



Chroma Vector

Mixer

- User
  - vol. 7.00
  - mute
- Piano
  - vol. 7.00
  - mute
- Agent
  - vol. 7.00
  - mute
- Piano
  - vol. 7.00
  - mute
- Synth
  - vol. 7.00
  - mute
- Metronome
  - vol. 7.00
  - mute
- Click
  - vol. 7.00
  - mute

Monitor

- Agent
- Inference
- Audio
  - rms



A digital piano keyboard interface with various controls. The keyboard is shown with white and black keys. Above the keys are various controls, including a volume knob, a mute button, and a metronome icon. The interface is dark-themed.



# Visual Components

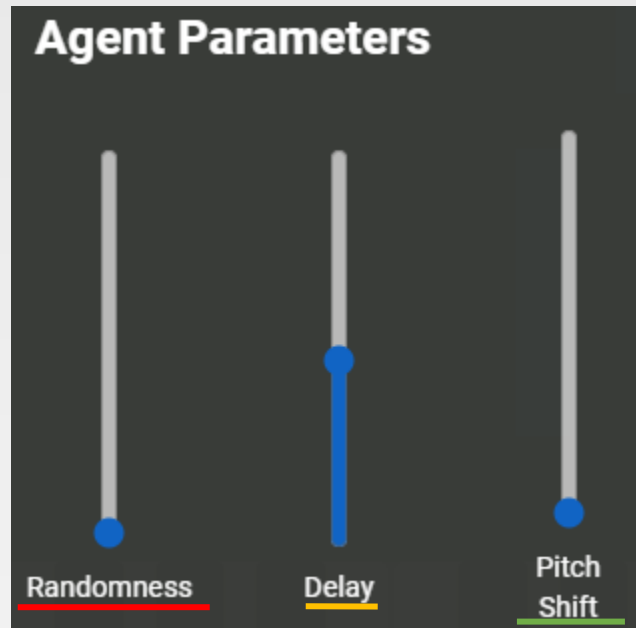
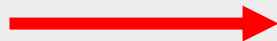
The image shows a music software interface with a settings dialog box overlaid. The background features a piano keyboard at the bottom, a piano roll on the left, and a MIDI piano roll on the right. A settings dialog box is centered, containing the following sections:

- Settings** (Title bar)
- Clock**: BPM (Max: Infinity) slider set to 100.
- MIDI**: Search field with the text "Type here to search for MIDI device".
- Agent Parameters**: Three sliders for Randomness, Delay, and Pitch Shift.

A red box highlights the word "Settings" in yellow text within the dialog box. In the bottom right corner of the interface, there are two yellow circular buttons: a play button and a settings gear icon.

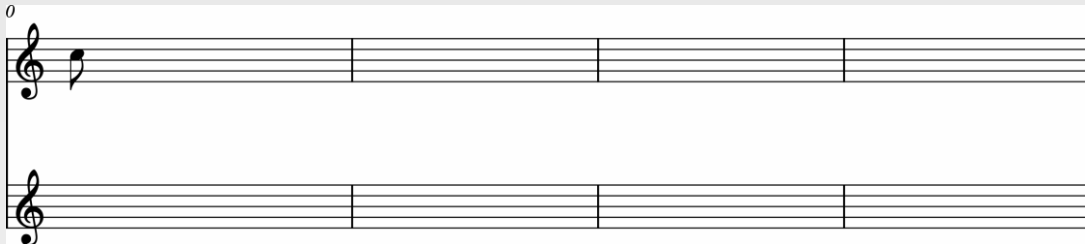
# Configuration

```
settingsModal:  
  sliders:  
    - id: 1  
      label: "Randomness"  
      value: 0  
      min: 0  
      max: 10  
    - id : 2  
      label: "Delay"  
      value: 8  
      min : 1  
      max : 16  
    - id : 3  
      label: "Pitch Shift"  
      value: 0  
      min: 0  
      max: 24
```



# Paradigms of Music Interaction

- Call & Response



# Paradigms of Music Interaction

- Call & Response

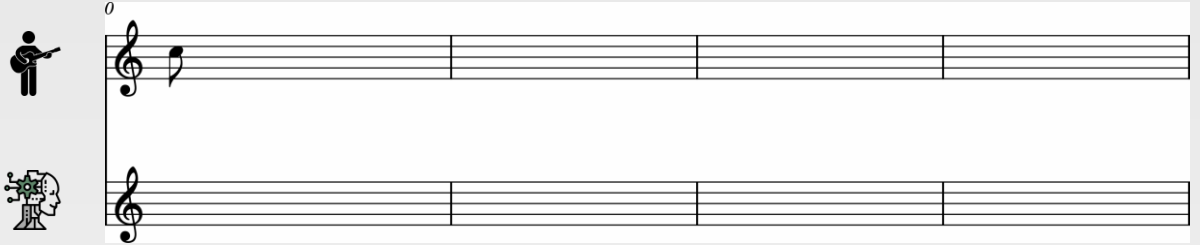


Diagram illustrating the Call & Response paradigm. It shows two staves of music. The top staff is marked with a '0' above the first measure and contains a single quarter note. The bottom staff is empty, representing the response.

- Simultaneous

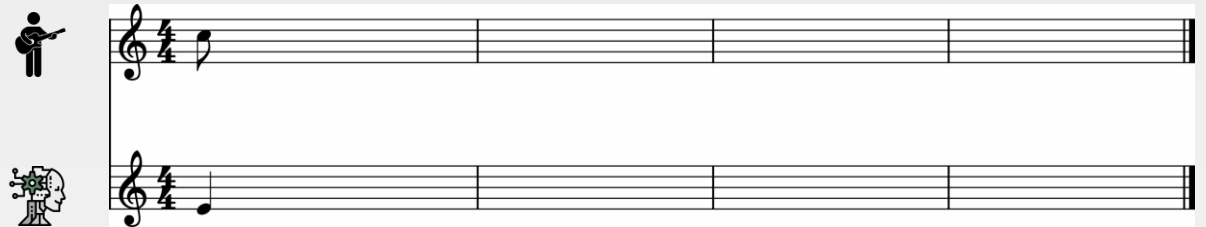
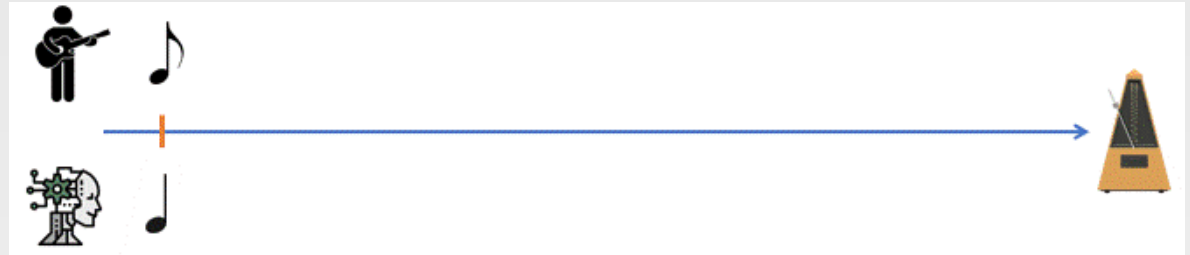


Diagram illustrating the Simultaneous paradigm. It shows two staves of music. The top staff starts with a treble clef and a 4/4 time signature, followed by a quarter note. The bottom staff starts with a treble clef and a 4/4 time signature, followed by a quarter note. Both staves have a double bar line at the end of the first measure, indicating simultaneous interaction.

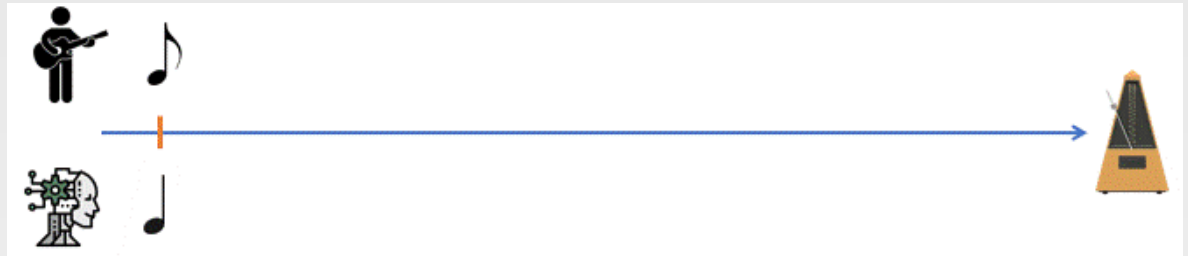
# Paradigms of Music Interaction

- Grid-based

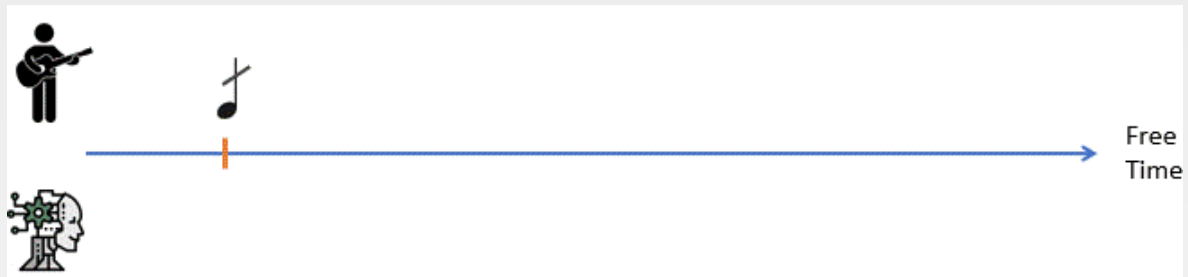


# Paradigms of Music Interaction

- Grid-based



- Event-based



# Configuration

```
title: "Euterpe"  
interactionMode:  
  noteMode: true  
  audioMode: false
```

```
noteModeSettings:  
  eventBased:  
    status: false  
  gridBased:  
    status: false
```

```
audioModeSettings:  
  windowSize: 1024  
  hopSize: 512  
  
clockSettings:  
  # ---- OPTION 1 --- #  
  # 16th-note grid on 4/4  
  ticksPerBeat: 4  
  timeSignature:  
    numerator: 4  
    denominator: 4  
  defaultBPM: 100  
  # ---- OPTION 2 --- #  
  clockPeriod: null
```

# Agent

- Provides 6 hook functions
  - **Empty** functions to be filled.
    - Invoked **automatically** at specific events or stages within the interaction
    - Can be activated/deactivated from the configuration file



```
function hook(event){  
  // your code  
}
```



# Euterpe Lifecycle

Initialization

Interaction



tick



Audio buffer



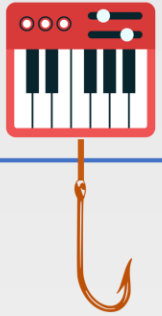
UI event



Audio buffer



tick



MIDI event

# Agent - Hooks

- `loadExternalFiles()`
  - Load external resources useful for the Agent
- `loadAlgorithm()`
  - Core algorithm initialization
  - Checkpoint fetching
  - NN model loading
  - warmup NN



# Agent - Hooks

- `updateParameter(id, value)`
  - Invoked when the user interacts with the GUI (buttons, sliders etc.)
  - The Agent's hyper-parameters are updated



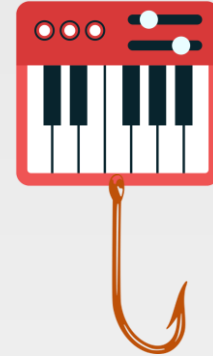
# Agent - Hooks

- `processClockEvent(tick)`
  - Invoked periodically based on the Clock's "tick"
  - Used on a time-grid based interaction



# Agent - Hooks

- `processNoteEvent(event)`
  - Invoked when a MIDI note is received
  - Used in “event-based” mode

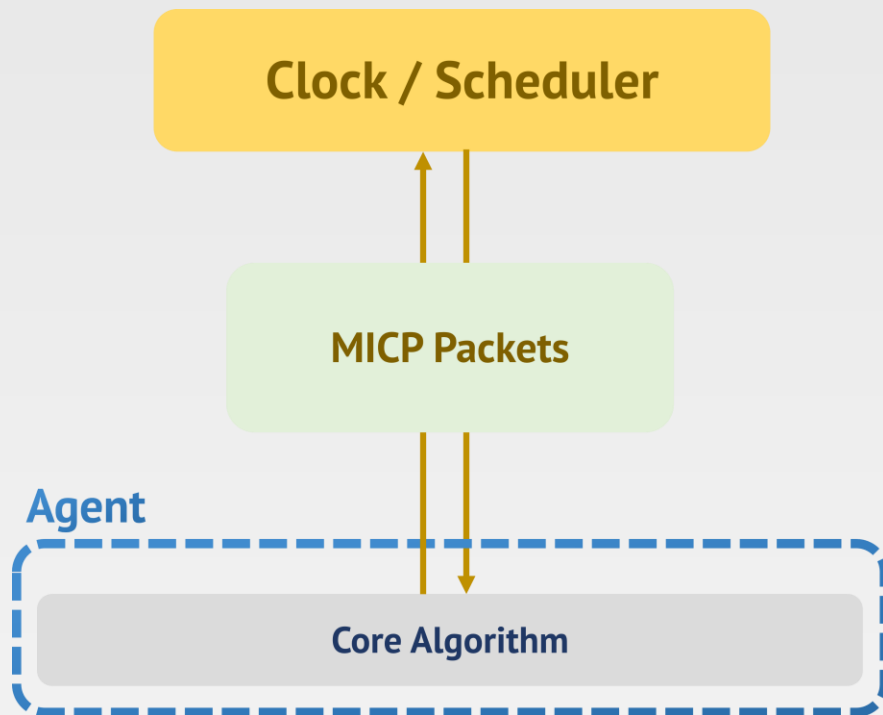


# Agent - Hooks

- `processAudioBuffer(buffer)`
  - Invoked when a new audio buffer is available
  - Every `hopSize` samples



# Music Interaction Communication Protocol - MICP



# MICP – NoteEvent

- **player** : Agent or User
- **instrument** : Which sampler instrument to use for playback
- **device** : The user's input device (i.e MIDI keyboard)
- **type** : Note\_On, Note\_Off or Note\_Hold
- **name/midi/chroma** : Info about the note (i.e C4, 60, 0)
- **channel/velocity** : Midi specific info
- **createdAt** (tick, seconds) : When was this note created/generated (timestamp)
- **playAfter** (tick, seconds) : Play the note with a delay
- **duration** : The duration of the note (optional)



# Coding Session

Online Guide :

<https://xribene.github.io/>