Integrating FM Synthesizer and Sequencer – Documentation

Dhathri Bathini

COMP 426: Digital Music Programming

Loyola University Chicago

**Project Overview:**

A useful feature for real-time performance could be the Interactive Sequencer in Max/MSP that is a real-time graphical patch to play, monitor and alter notified musical sequences. This project consists of three different note sequences, represented in coll objects, each with data about pitch, velocity, and duration. Metro, counter, sel, and gate objects allow users to switch cycling automatically or follow program sequences depending on the user’s preference. This one comes with virtual multi-slider control to visualize sequence data during play; zl.scramble introduces random variations every three bangs for variation. Also, a live interface for adjusting notes’ values is implemented, providing great control over the sheet’s playback performance. MIDI notes are created with the makenote object and sent via noteout for audible signals so the sequencer can work for timing, interaction or audition.

In addition to this, the FM Synth project further increases modularity and expand the functionality of the instrument through integration of the FM synthesizer that makes it possible to produce sound dynamically. The synthesizer takes information from the sequencer such as MIDI note and velocity data to frequency modulate the carrier and modulating oscillators to produce complex sounds with depth. Of thispoly~, it has an ADSR envelope for controlling the dynamics of sound and can play multiple voices at the same time. When combined, it transforms the sequencer into a full-fledged platform that controls real-time musical performance and sound design in a single interface.

**Features:**

1. **Frequency Modulated Synthesis:**

The synthesizer has the feature of frequency modulation to develop rich and variant sound. Two oscillators are considered here, a carrier oscillator and a modulating oscillator whose frequencies are determined using controllable parameters such as harmonicity ratio and modulation index. This interaction helps create rich sound blends which enrich sound design opportunities.

1. **ADSR Envelope Control:**

adsr~ controls the dynamics of a sound sample and grants the possibility of enacting the attack, decay, sustain, and release times. This feature adds expressiveness by allowing the user to form the way the sound develops temporally, according to MIDI velocity for additional regulation.

1. **Polyphony Management:**

The thispoly~ object moves between for multiple active voices simultaneously, thus allowing smooth polyphonic implementation. It makes it possible to control several concurrent notes to avoid hitches in audio sequences and guarantee nice transition.

1. **Real-Time Parameter Control:**

This synthesizer includes actual time manipulation of factors like carrier frequency, harmonicity ratio, as well as modulation index. These parameters can be changed by the users using the sliders to make dynamic and aesthetic changes during the live performances.

1. **Flawless integration of Sequencer:**

Designed to synchronize with the Interactive Sequencer tightly, the synthesizer further receives the MIDI note and velocity data from the sequencer and upon processing, generates a modulated output sound. That is, this integration enhances the extendibility of the system which has caused the enhanced sound synthesis along with the flexible sequence play.

1. **MIDI Compatibility:**

The synthesizer works with standard MIDI and accepts external MIDI sources and other DAWs for MIDI input. It also handles note\_on/note\_off, velocity scaling, and an mtof function for converting pitch to frequency for proper integration of the library with other hardware and software tools.

**Patch Structure:**

1. **MIDI Input Processing**

* MIDI Note and Velocity Input:
  + The synthesizer abstraction accepts a list of two integers: MIDI note number and velocity are the two components in this case.
  + The pitch is then changed to frequency with the help of mtof while velocity is normalized to control amplitude of the note.

1. **FM Synth Engine**

* Carrier and Modulating Oscillators:
  + A cycle~ object produces the carrier frequency, which is amplitude-modulated by another cycle~ object controlled by the harmonicity ratio and modulation index.
  + They tailor the modulator amplitude using user parameters and thus have a dynamic FM synthesis.

1. **ADSR Envelope Generation**

* Envelope Control with adsr~:
  + The adsr~ object is used to specify the attack, decay, sustain, and release phases of the sound as the object is triggered by MIDI note – on events.
  + MIDI velocity is utilized to modulate the amount of the ADSR envelope for dynamic variations in sound character.

1. **Polyphonic Voice Management**

* Voice Allocation with thispoly~:
  + This thispoly~ object monitors the active voices and controls their allocation and release in the course of polyphonic execution.
  + Copes with doubling of notes well enough so that there are no sudden fade outs or jumps.

1. **User-Controlled Parameters**

* Real-Time Adjustments:
  + There are three control knobs which allow the user to set the carrier frequency, harmonicity ratio as well as the modulation depth.
  + These parameters control acute characteristics of the synthesized sound and are perfect for real time soundscaping.

1. **MIDI Note Output Integration**

* Output to Built-In Synth:
  + Through MIDI, a sequencer sends pitch and velocity values from the sequencer’s notes to the synthesizer’s ins.
  + Note-off events occur to release the announced sound by freeing the ADSR envelope.

**User Guide:**

1. **Activating Playback:**

As a start, the synthesizer has to be connected to the sequencer to begin sound generation. Turn on the playback button on the top left corner of the sequencer. This creates the metro object to start the bangs at specified time or in the default of 500ms and feeds them to the counter, which spins through the indices to play sequences in the coll objects. The output is the sent the synthesizer for the actual sound production.

1. **Sending MIDI Notes to the Synthesizer:**

It generates MIDI pitch and velocity data which are then forwarded to the synthesizer’s abstraction. Make sure that the synthesizer abstraction is in poly~, waiting to parse note information.

1. **Shaping the Sound with ADSR Envelope:**

Continuously manipulating the four main parameters on the synthesizer interface, namely attack, decay, sustain, and release (ADSR). For instance:

Attack: Decide how soon the volume increases to its highest level.

Decay: The time taken to get to the sustain level has to be described.

Sustain: Adjust the level of the sound when its variability is low.

Release: Define how fast the sound disappears after the note-off event has occurred.

1. **Controlling Polyphony:**

Thus, the synthesizer is capable of polyphonic play back. Use the sequencer tool or another MIDI controller to play several intersecting notes to hear how it adds layers to the sound; thispoly~ maintains clear voice allocation without any crackles.

1. **Adjusting Playback Speed:**

Alter the position of the metro object on the sequencer to change note duration. Faster interval generates fast sequences of notes and slows down when the intervals are slow hence changing the time it takes for the synth to produce sounds.

1. **Hearing the Output:**

Make sure that the audio output of the computer is set in the Max/MSP environment. The synthesizer sends out its processed sound through the dac~ object. Headphones or external speakers should also be used for best sound quality.

1. **Introduction to Randomization:**

The second step is to engage the multi-slider in the sequencer where you can see the current active sequence. Additional features can be turned on by opting for zl.scramble which causes the values of the sequences to play randomly so as to create different and unexpected sounds.

A screenshot of a computer

Description automatically generated

A diagram of a sound system

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated