I initially used sinusoidal frequency waves.

That sounded bad.

FluidSynth

A synthesizer.

- FluidSynth allows nodes in MIDI form to be played through SoundFonts (a public collection of sounds)
- Thus, to play sounds: a note to midi conversion was necessary

```python
def play_chords(chords, duration=0.1, velocity=100):
    current = set()
    for chord in chords:
        print("Now playing:", chord)
        next_notes = set(filter(None, (note_to_midi(n) for n in chord)))
        for note in current - next_notes:
            fs.noteoff(0, note)
        for note in next_notes - current:
            fs.noteon(0, note, velocity)
        time.sleep(duration)
        current = next_notes
    for note in current:
        fs.noteoff(0, note)
```

Direct Music Generation Code

```python
# -------------------- MIDI PLAYBACK --------------------
def note_to_midi(note):
    if note in ["00", "0", "", None]:
        return None
    try:
        note = note.replace('*', '')  # Handle natural sign as a no-op

        # Expect format like "D4b" or "C4#"
        if len(note) < 2:
            return None

        pitch_letter = note[0]
        if note[2:] in ('#', 'b'):  # with accidental
            octave = int(note[1])
            accidental = note[2]
        else:
            octave = int(note[1])
            accidental = ''

        pitch = pitch_letter + accidental

        # Convert flats to equivalent sharps
        flat_to_sharp = {
            'Cb': 'B', 'Db': 'C#', 'Eb': 'D#', 'Fb': 'E',
            'Gb': 'F#', 'Ab': 'G#', 'Bb': 'A#'
        }
        pitch = flat_to_sharp.get(pitch, pitch)

        note_map = ['C', 'C#', 'D', 'D#', 'E', 'F',
                    'F#', 'G', 'G#', 'A', 'A#', 'B']

        return note_map.index(pitch) + 12 * (octave + 1)
    except Exception as e:
        print(f"Error converting note '{note}': {e}")
        return None
```