Fork: Live-Composing a Soundtrack for Chess

Victor Zheng

North Central College
vzheng2696@noctrl.edu

ABSTRACT

Fork is a music composition algorithm that composes a soundtrack using data derived from chess games. Unlike traditional sonification, Fork devises a musical grammar that combines modal and tonal elements into a flexibly functional preparation and resolution system, ensuring a sense of musical progression that mirrors the natural progression of a chess game. It uses SuperCollider to efficiently control random parameters that control note output as well as sonic parameters in synthesis, applying both to produce musical development.

1. DATA INTERPRETATION

Fork depends primarily on tracking the threats and captures initiated between pieces. Threats are represented by the variables `wAtk` and `bAtk` for threats initiated by White and Black respectively. At each move, the total between `wAtk` and `bAtk` is compared with the total value from the previous move, represented by the variable `atkDel`.

The *stability* parameter reflects the dynamic state of the game, derived from the presence of captures or checks. The presence of either will set stability (`stab`) at 0. Any moves that do not cause either will increase `stab` by 1. `stab` is initialized at 3 at the beginning of a game.

Below is the opening of a game illustrating the behavior of the various parameters at each move.

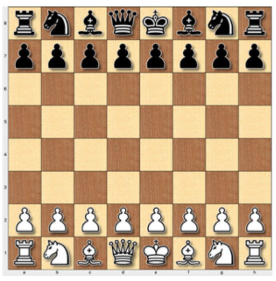

Figure 1. Starting position.

```
wAtk = 0;
bAtk = 0;
atkDel = 0;
stab = 3;
```

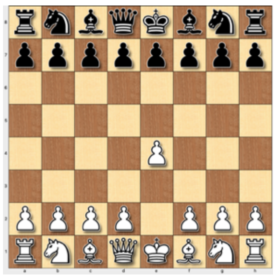

Figure 2. 1. e4

```
wAtk = 0;
bAtk = 0;
atkDel = 0;
stab = 4;
```

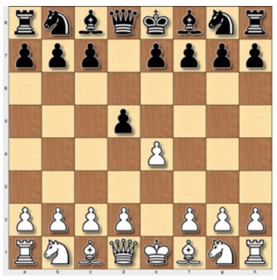

Figure 3. 1...d5

```
wAtk = 1; //d5
bAtk = 1; //e4
atkDel = 2;
stab = 5;
```

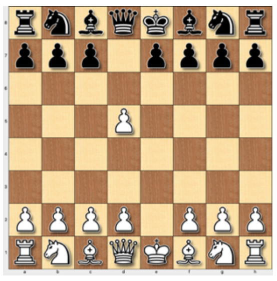

Figure 4. 2. exd5

```
wAtk = 0;
bAtk = 1; //d5
atkDel = 1;
stab = 0;
```

2. MELODIC COMPONENTS

Fork translates the aforementioned parameters into music through several musical components, structured around a Dorian scale.

The primary musical component is the *direct tone*, a tone which plays at the moment of every move. Each direct tone is derived from a note randomly chosen using primarily the atkDel variable:

- When atkDel is equal to 0, the possible tones comprise the i triad to represent the original tonic. In C Dorian, this translates to the notes C, E♭, and G, each with an equal chance of being selected. Musically, this portrays a stable state as the starting tonic.
- When atkDel is equal to 1, the possible tones are the first two notes of the ii triad, corresponding to D and F in C Dorian. This produces an unstable musical state, where the draw is to return to tonic.
- Finally, when atkDel is 2 or higher, or when a capture or check is taking place, the possible tones correspond to the VII triad, or B♭, D, and F in C Dorian. The introduction of these notes also produces an unstable musical state, but with a draw to establish a new stability away from the original tonic.

The use of the Dorian scale avoids the leading tone to the tonic and exploits the half step leading from D up to E♭ as well as from A♭ down to G, allowing for more flexible possibilities in resolution through these auxiliary tendency tones while sacrificing tendency towards C, with the rationale that its preponderance from stable direct tones will lend it enough tonic weight on its own.

Each direct tone is fed into a list, bgNotes. bgNotes provides the source notes that comprise the *background melody.* The notes play intermittently at first, and then coalesce into a continuous pattern of 16th and 8th notes after the 6th move is played. This staggering of entries of different musical entities serves as a way to control musical progression that reflects the game state.

The background melody plays randomly generated notes taken from bgNotes. Repetition is allowed in bgNotes; long stretches of atkDel equal to 0 will therefore steer the melody towards a C minor sonority as C, E♭, and G propagate through the list, while more consistently higher atkDel values will pull the melody towards a B♭ major sonority. bgNotes has a maximum size of 10; any further additions after it attains size 10 will eliminate the oldest value of the list, thereby representing only the previous 10 moves.

Concurrently, the *high melody* begins playing after the first capture is made. Unlike the background melody, which starts after a set number of moves, the high melody can begin earlier or later depending on how conservatively the game is played, allowing for some variance in musical progression. The high melody also uses the notes from bgNotes; rather than using a random distribution for the notes, the high melody reproduces bgNotes in exact order, with each note having a chance to be raised one or two octaves. The high melody uses a similar weighted random distribution for rhythm, using the same random weights as the background melody, but slowly and with long pauses rather than playing continuously.

3. HARMONIC COMPONENTS

In harmonic support are four main components, a *bassline,* a *chord*, a *bass rhythm*, and a *high percussion*. The bass line is a single tone that plays continuously throughout the game. It begins on scale degree 1, C in C Dorian. The *chord* is formed by up to 16 continuous tones randomly distributed across different octaves of the triad formed from the bass line. The 16 tones are initialized as silent, with one being activated with every two captures. The effect is subtle at first, but as the game progresses with mounting captures, the bassline and chord will become more and more prominent, producing a large-scale developmental trajectory that manifests over the entire game.

The bassline and chord shift simultaneously with the direct tone when a capture or check is initiated, resetting stab to 0 and producing a harmonic shift. This represents a *progression*, where harmony moves in a way that departs stability. Each chord has a number of possibilities for unstable movement, represented in Figure 5 below:

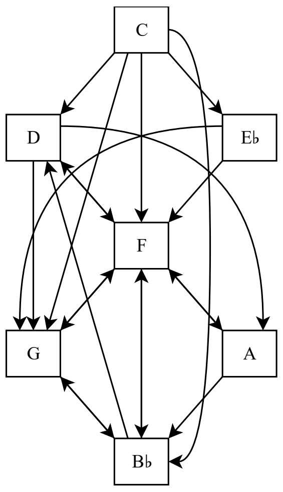

Figure 5. Progression map. All chords have one or more choices in progressing. Progression paths are reminiscent of tonic to predominant motion in common practice harmony, primarily avoiding leading tone dominant-tonic motion except when reaching B♭, which are treated by the direct tone and melody as inherently unstable.

When stab exceeds 3, thus returning the game to a stable state, the chord will *resolve*, following its own set of possibilities, represented in Figure 6.

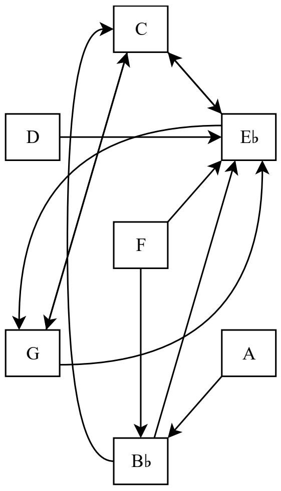

Figure 6. Resolution map. Far fewer choices are available with resolution than with progression, using the C, E♭, G, and B♭ chords as possible destinations. Leading tone dominant-tonic motion is more emphasized in cases of B♭ and E♭, but the inclusion of more indirect resolution paths such as C to E♭ and back maintains a balance between the four possible resolution endpoints.

The bass rhythm begins playing after the first capture, when the game settles into its first stable state after opening exchanges. The bass rhythm takes the existing note being played in the bass line and arpeggiates notes in random order from the triad forming from the bass note in accordance with the scale, thus producing a broken chord to accompany the continuous bass and chord tones. The rhythm is a steady 16th note pattern with an accent every four notes, thus producing a beat every quarter note. The bass rhythm begins panning back and forth in a similar manner to the background melody and the high melody after the first resolution.

The high percussion begins playing intermittently after the first resolution. Similarly to the high melody and background melody, its specific entry condition contributes to the forward progression of the music and its entry depends on the nature of play. The high percussion component comprises the bass note and a third up, both elevated by two octaves. The high percussion pattern involves repeating the two notes in a random rhythmic pattern comprising 80% sixteenth notes and 20% eighth notes, producing a rapid pattern that falls in time with the background melody and bass rhythm.

4. HIERARCHY OF CONTROL

The hierarchy of note generation is therefore an integral aspect of Fork's musical generation and output. The direct tone is the main immediate output to each move and the only element directly beholden to the players' input. Other effects down the hierarchy present as more indirect effects, manifesting asynchronously to the direct input. Changes to the music thus propagate through the hierarchy to produce a range from immediate to more gradual change across the trajectory of the entire game, applying a level of independence to the musical development rather than being fully beholden to the players' playing rhythm. This propagation is presented below in Figure 7.

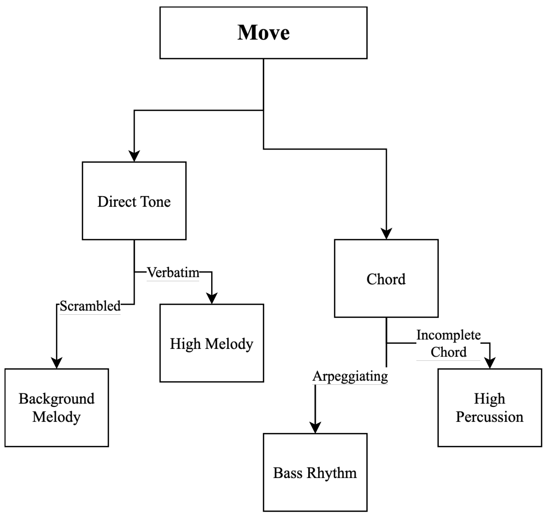

Figure 7: Hierarchy of control. A player does not need to alter any aspect of their chess-playing technique to control the soundtrack compellingly. Their move is their only control vector on the soundtrack, but its effect on the music is propagated through a hierarchy to control an ultimately deep, diverse range of musical parameters.

Figure 8 shows in score form a sample realization of the notes that might result from the opening shown above. For demonstration purposes, the bass rhythm is shown as playing continuously throughout and the high percussion, background melody, and high melody are shown as realized at the end of White's 5th turn, all disregarding the usual conditions for them to begin playing.

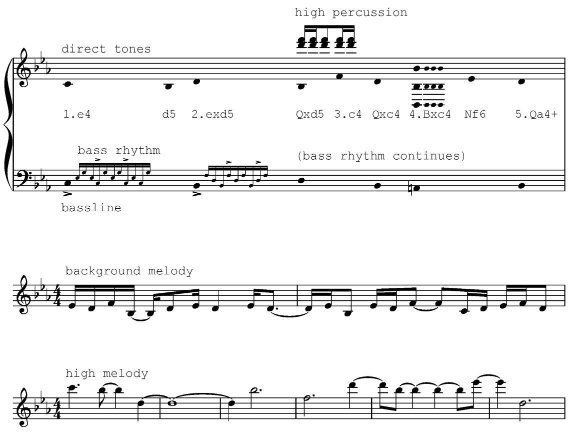

Figure 8: Sample realization.

5. SYNTHESIS

Three main SynthDefs produce the sounds that comprise the total output.

```
(
  SynthDef(\bass, {
     arg freq, pan = 0, noise = 0.01, amp
= 0.2, ampLag = 2, sus, out = 0;
     var sig;
     sig = 15.collect({
     LFSaw.ar(freq*rrand(1 - noise, 1 +
noise), mul:0.05);
     });
     amp = amp.lag(ampLag);
     sig = Splay.ar(sig,
LFNoise1.ar(1).range(0.5, 0.9));
     sig = MoogFF.ar(sig,
LFNoise1.ar(0.2).range(550, 1000));
     sig = sig.scramble;
     sig = sig*amp;
     Out.ar(out, sig);
}).add;
);

(
  SynthDef(\rhythmicBass, {
     arg freq, pan = 0, noise = 0.01, amp
= 0.2, out = 0;
     var sig, env;
     env = EnvGen.ar(Env.perc(), doneAc-
tion:2);
     sig = 15.collect({
     LFSaw.ar(freq*rrand(1 - noise, 1 +
noise), mul:0.05);
     });
     sig = Splay.ar(sig, center:pan);
     sig = MoogFF.ar(sig,
LFNoise1.ar(0.2).range(550, 1000));
     sig = sig.scramble;
     sig = sig*env;
     sig = sig*amp;
     Out.ar(out, sig);
  }).add;
);

(
  SynthDef(\moveTone, {
     arg freq, pan = 0, noise = 0.1, amp =
0.2, atkTime = 0.01, susTime = 0, relTime
= 1, out = 0;
     var sig0, sig1, sig, env;
     env = EnvGen.ar(Env.new([0, 1, 1, 0],
[atkTime, susTime, relTime], \sqr), done-
Action:2);
     sig0 = 5.collect({
     SinOsc.ar(freq*rrand(1-noise,
1+noise), mul:0.01);
     });
     sig0 = Mix.ar(sig0);
  sig1 = 5.collect({
     LFSaw.ar(freq*rrand(1-noise,
1+noise), mul:0.01);
     });
     sig1 = Mix.ar(sig1);
     sig = Mix.ar([sig0, sig1]);
     sig = sig * env;
     sig = Pan2.ar(sig, pan);
     sig = sig*amp;
     Out.ar(out, sig);
  }).add;
);
```

All tones that produce the bassline and the chord components are generated by the `\bass` SynthDef. The `\rhythmicBass` SynthDef is identical in synthesis to \bass, only with an integrated envelope that allows it to perform the bass rhythm musical component instead of the continuous tones found in the bassline and chord. Finally, the `\moveTone` is responsible for producing the direct tones, background melody, high melody, and high percussion components. Apart from the continuous bass and chord tones, which are simply continuously running Synths, all patterns are rendered using Pbindefs which run continuously, having their arguments adjusted with each move.

Most musically significant is the noise argument, present in all three SynthDefs. As each SynthDef employs `.collect` to stack a number of waveforms with the same base frequency, noise introduces a random amount of detune to each one, resulting in a choral effect at low values and noise effect at high values. In the short term, this timbral variance maintains a level of dynamicism with each note through the subtle choral, beating timbre. Noise becomes programmatically significant as the cumulative noise from the stacked tones in the chord component gradually transforms the clear sonority into a less resonant, fuzzy effect as chord tones enter.

6. CONCLUSIONS

The translation of human input, i.e. the chess game, into musical structures, can inform an unprecedented degree of interactivity in video game music composition. Fork explores the ways in which both the input and output of such information exchange can be quantized, and in the case of the musical output, dives into parameters beyond that of traditional note-based music theory that allow quantization of sonic parameters beyond the constraints of any musical system.

Fork employs controlled randomness to ensure that each rendering, even of the same exact game, produces a unique musical result. This uniqueness is crucial to the artistic vision of the project, to avoid any possibility of a 1:1 conversion between a game and a specific output.

The subtle yet continuous movement in specific sonic parameters controlling spatialization and timbral control set Fork apart from traditional MIDI note generation. By controlling the synthesized sound directly, it transcends the boundaries of traditional note composition and expands the possibilities of algorithmic control beyond that of traditional music theory.