Vol.2016-MUS-112 No.8
2016/7/31

DJCoder: DJ システムと密に連携したプログラミング環境

原健太[†1]　加藤淳[†2]　後藤真孝[†2]

概要：本稿では，DJ 機器および DJ システムをプログラミングで制御可能にすることで，コンピュータと人間が共同でミックスを行う DJ プレイ手法を提案する．ユーザは，事前にプログラミングしておくことで，つまみを 6 つ高速かつ正確に同時に動かすなど，人間には難しい制御を披露できる．さらに，即興でプログラミングすることも可能で，その場の雰囲気に合わせた選曲変更に対応したり，プログラムのパラメタを調整したりして，コンピュータとの B2B プレイ（Back-to-back; 2 人の DJ が交互に選曲する協力プレイ）ができる．

キーワード：DJ, B2B, プログラミング環境, Live Programming

1. はじめに

DJ は曲と曲を繋げて（これをミックスと呼ぶ）音楽を止めないパフォーマンス手法である．プレイ手法の１つに，Back2Back(B2B)という２人で交互に楽曲を掛けあう DJ プレイ手法がある．B2B 中には２人でディレイやフランジャーなどのエフェクト(FX)や，イコライザ(EQ)を同時に操作するなど，１人では出来ない特殊なプレイが行われることがある(図 1)．本研究ではこの B2B に着目し，DJ 機器および DJ システムをプログラムで制御可能にすることで，コンピュータと人間が共同でミックスを行う DJ プレイ手法およびそのためのプログラミング環境「DJCoder」を提案する．DJ のつなぎをプログラミングできれば，コンピュータが DJ プレイをしている途中に，エモーショナルな FX の操作を人間が行うといった，B2B のような表現力の高い DJ プレイを行うことができる．

DJ は現場で選曲を変えたり，曲の中で繋ぐポイントを変えたりしたい場合がある．そこで本研究では，DJ システムとプログラミング環境を密に連携させ，DJ プレイ中にプログラムを変化させられる Live Programming を可能にした．また，DJ プレイはタイミングが重要なため，タイミング制御のための API を設計し，DJ システムの精緻な制御を実現した．

2. 関連研究

2.1 プログラムによる楽曲制作・パフォーマンス支援

楽曲制作・パフォーマンスを可能にするプログラミング環境は数多く提案されている．SuperCollider[1], ChucK[2], Max/MSP[3]および Tidal[4]は，いずれも音響処理を行うプログラムを作りながら即興で音楽を演奏できる Live Programming 環境である．しかし，DJ プレイに着目したものではないため，DJ 特有の小節区切りでのタイミング制御が難しい．Tidal はループ構造を伴う音楽のタイミング制御を直感的に行える API を提供しているが，DJ プレイではループするようなパターンは少なく，そのまま流用することはできない．そこで我々のシステムでは，テンポ情報を元に小節区切りで動作する関数や，ミックスの展開とミックスの手法を表現しやすくするオブジェクトなど，DJ に特化したタイミング制御の API を設計した．

図 1 Back2Back で DJ をしている様子 [a]

プログラミングを用いない DJ プレイ支援システムにも既存の取り組みがある．Hirai らの MusicMixer[5]では，楽曲のビートなどの自動解析情報を元に，繋がりやすい楽曲をクロスフェードで自動ミックスする．Algoriddim 社[6]の DJ システム djay シリーズには automix 機能が搭載されており，楽曲解析結果に基づいて繋がりやすい曲を選択，予め決められたミックス手法の中からランダムもしくはユーザが選択した方法で自動ミックスを行うことが可能である．我々のシステムはミックスをコンピュータに行わせるという点で類似しているが，ミックスの手法を自分でプログラミングでき，自分の想像したミックスを自由自在に制作できるという点で異なる．

2.2 既存アプリケーションの拡張・自動化手法

本研究では，後述する事前取材の結果をもとに，DJ プレイに利用されるソフトウェアの使い勝手を損なわずに拡張

†1 明治大学
Meiji University
†2 産業技術総合研究所
National Institute of Advanced Industrial Science and Technology (AIST)

a glenjamn3 氏による DJ KORE & BRODINSKI - BEATS BY BROS @ BROMANCE GRAND CENTRAL MIAMI - 3.24.2014 ライセンスは CC BY 2.0 に基づく

し，プログラミングで自動化する手法を提案する．このように，既存アプリケーションの動作を拡張・自動化する手法はエンドユーザプログラミングの文脈で多く提案されてきた．例えば，既存 Web アプリケーションの GUI の一部をセルとして切り貼りして新しい GUI を構築できる手法[7]や，GUI のスクリーンショットをソースコードエディタに貼り込めるプログラミング環境 Sikuli[8]などがある．

これらの既存研究では，汎用性を重視して通常の GUI 操作とメタな拡張動作を明示的に分けている．一方，本研究では，DJ システムに特化したタブインタフェースを追加し，通常の使用とプログラミングによる拡張をスムーズに行き来できるインタラクションを実現した．

3. DJ 経験者への事前取材

インタラクションデザインするにあたり，形式張らない形で DJ 経験者にヒアリングしヒントを得た．DJ 経験者 3 人にアナログ DJ（アナログレコードを用いた DJ），CDJ（コンパクトディスクを用いた DJ），PCDJ（パソコン上のソフトウェアと MIDI コントローラを用いた DJ）の利用経験とその感想，B2B の経験，DJ プレイ中の曲と曲のミックス方法についてインタビューした．

3.1 各 DJ 機器の経験とその感想

- DJ 歴 3 年の A さんは，アナログ DJ，PCDJ，iPad を用いた DJ を経験している．現在は iPad を用いて DJ をしており，手軽さが気に入っているものの，操作性や表現力の乏しさから，数年前に使用していた MIDI コントローラを使った PCDJ に戻りたいと回答した．
- DJ 歴 1 年の B さんは，MIDI コントローラを使った PCDJ のみの経験である．PCDJ システムがテンポ合わせを行ってくれる部分が気に入っていると回答した．
- DJ 歴 2 年でイベントのオーガナイズ経験もある C さんは，PCDJ と CDJ を経験している．自宅では MIDI コントローラを併用した PCDJ を利用しており，手軽な部分が気に入っていると回答した．一方現場では，PCDJ だと PC を見続けなければならないことでパフォーマンスが思ったように行えないと考え，PCDJ よりも CDJ を好んでいると回答した．

これらのインタビューから，iPad が手軽だとしても音楽的表現力がネックとなり PCDJ に戻ることを検討していたり，PCDJ が手軽だとしてもパフォーマンスの面での表現力がネックで CDJ を利用していたりと，何か別の魅力が合ったとしても，自分の目指す表現が行えない場合には環境を変える人が存在していることが分かった．

DJ をプログラムするという新たな手法を DJ プレイに導入するにあたって，PCDJ の利便性や表現力は損なわないように気をつけてシステムをデザインする必要がある．

3.2 B2B についての感想

- 一度 B2B を行ったことのある A さんは，相手がどんな選曲をしてくるのかの「ドキドキ感」が楽しいと回答した．相手が選曲中の空き時間にパフォーマンスをして会場を巻き込んだり，会場の様子をいつもよりも眺めることができたりすることが B2B の利点であると回答した．
- 一度 B2B を行ったことのある B さんは，自分の趣味嗜好に合った新しい楽曲を知ることが出来る点が良いと回答した．
- 7 回以上 B2B を行ったことの C さんは，一緒にプレイしている相手が自分の DJ プレイをフォローしてくれる良さがあると回答した．自分では選ばない選曲を相手がしてくることが面白いと感じており，相手がミックス中にエフェクトを入れたりするなど，2 人でないと出来ないプレイができるところは面白い部分であると回答した．

これらのインタビューから，B2B 経験者は自分だけでは思いつかない選曲や，プレイから解放される空き時間が生まれる点，複数の操作を同時並行で行える点など，「1 人では出来ないプレイ」に魅力を見出していることが分かった．

3.3 曲のミックス方法

- A さんはジャンルに合わせて繋ぎ方を変えることが多いと回答した．サビでさり気なく曲を入れ替えるパターンや，EQ のロー（低域）に相当する周波数帯域を先に入れ替えておき，その後，全周波数帯域において楽曲を入れ替えるパターンを多用すると回答した．
- B さんは事前に再生箇所，入れ替える部分などにキュー（再利用可能な目印）を打ち，当日はそのキューに沿って DJ ミックスを行うと回答した．フィルターや FX などは多用せず，クロスフェードを多用すると回答した．
- C さんはトランスなどの楽曲には定石があると回答した．曲をサビまで聴き終わって最後の部分で次の曲を再生するなど，曲の区切りに着目して DJ ミックスを展開させていくと回答した．おおまかに楽曲にあたりをつけ，展開するタイミングは楽曲を聞きながら決める．縦フェーダー（音量調整器）を操作して楽曲を交互に入れ替えながら DJ ミックスを行うことで，曲が次に変わるという雰囲気を演出していると回答した．

これらのインタビューから，ある程度 DJ プレイを行ったことのある中級者には DJ ミックスの「パターン」が存在していること，ジャンルに対して適切なパターンがあることが分かった．

3.4 実装指針

以上の取材結果を元に，次のような指針を立てて提案シ

選曲モード(Traktor Pro 2フル機能)　　コーディングモード

図 2　DJCoder の画面構成．選曲モードの画面(左)，コーディングモードの画面(右)．画面下部の左側のタブでモードを切り替えることができ，選曲モードでは DJ システム Traktor Pro 2 にフルアクセスできる．

ステム DJCoder を実装することにした．

- 「1 人では出来ないプレイ」や「人間には操作が困難なプレイ」を実現するために，DJ プレイをプログラマブルにするシステムを開発する．
- PCDJ の利便性や表現力を維持するために，市販の DJ システムに別アプリケーションをオーバーレイする形で実装し，DJ システムの機能をフルで活用できるようにする．
- DJ プレイのプログラムを単純化するため，PCDJ のテンポ合わせ機能は活用する．
- 事前にプログラムはある程度準備して本番に望むことを前提に置いて開発する．

4. インタラクションデザイン

本章では，事前取材をもとに設計した DJCoder のインタラクションデザインについて述べる．

4.1 動作環境

図 3 動作環境の一例
(左: DJ コントローラ 右: コンピュータ)

図 3 に DJCoder 動作環境の一例を示す．図左は DJ コントローラで，右はコンピュータである．ユーザは基本的にコンピュータでプログラムを編集・実行したりして DJ プレイを行う．プログラム実行時には DJ コントローラを使って FX を適用するといった通常の DJ 操作を追加で行うこともできる．

4.2 画面構成

本システムは，市販の DJ システムにオーバーレイ表示される．図 2 に示すように画面上部は DJ システムのデフォルト画面そのものであり，現在再生されている楽曲情報や，ミキサーやエフェクトの情報が表示されている．ユーザは，画面下部の左側にオーバーレイされたタブで，DJCoder の主要機能であるコーディングモードと，DJ システムの選曲モードをいつでも入れ替えることができる．

コーディングモードでは，画面下部にエディタエリアとグラフエリアが表示される．グラフエリアには，プログラムで後述するパラメタ制御オブジェクトを使った場合に値の推移がグラフで表示され，おおまかな値の推移が確認できる．エディタ下部の「Run」ボタンを押すとエディタに入力したプログラムが実行される．

4.3 DJCoder のユーザ体験

4.3.1 準備フェーズ

DJ は本番前に選曲を行ったり，ジャンル別に楽曲をフォルダ分けなどして楽曲の整理を行ったりすることが多い．楽曲がある程度絞られたら，同時に DJCoder を用いてプログラミングも行う．ユーザが利用できる DJ プレイのための API に関しては 4.4 で述べる．

4.3.2 本番フェーズ

プレイ中のユーザの動作は次の動作を繰り返すものになる．

1. 曲を選んでセットする．
2. 任意のプログラムを読み込み，プログラムの変数などを変更してミックスをアップデートする．
3. 任意のタイミングでプログラムを実行する．
4. プログラムが動作する．並行してコントローラを使って操作することもできる．

なお，DJ システムと自作ソフトウェアのタイミング同期を行うために，一曲目の再生後，四拍目と次の一拍目の間で SYNC ボタンを押して小節の頭出しを行う必要がある．SYNC ボタンは拍ごとに点滅しており，一拍目に限っては

図 4 実行中のミキサー部分のズーム画像．プログラムが操作している部分にはパーティクルが表示される ．

色が変化するため，視覚的にテンポの同期がずれているかを確認できる．

なお，本システムはあくまで DJ システムの制御をプログラムで行える追加機能であり，ユーザは追加機能を一切使わず普段の DJ プレイを行うこともできる．

プログラムの実行中には，プログラムが操作している部分にパーティクルが飛んでおり，どの部分をプログラムが操作しているのかを視覚的に認識することができる(図 4).

4.4 DJ プレイのための API

DJ プレイのプログラミングは JavaScript で行う．DJCoder では，DJ プレイに固有の機能を提供するために 4 つの API（Application Programming Interface）を提供する．

4.4.1 *decka, deckb*: Traktor 制御用オブジェクト

decka, deckb は Traktor を制御するためのオブジェクトである．***decka*** が A デッキ，***deckb*** が B デッキを示しており，Traktor への操作はすべてこのオブジェクトを介して行う．操作可能なパラメタを以下に示す．

- 各デッキのゲイン・縦フェーダー・イコライザ(高域・中域・低域)
- クロスフェーダー(A デッキと B デッキの混合割合)
- 各デッキに接続された 3 つの FX のオン・オフ・エフェクト切り替え・パラメタ調整
- 各デッキの再生・停止

例として，A デッキを再生し，同時に 1 つめのエフェクトをオンにするプログラムを示す．

```
1 decka.play();
2 decka.fx.a.on();
```

4.4.2 タイマーオブジェクト

DJ プレイをプログラミングする際にはタイミングの制御が必須である．そこで，時間制御を行うための機能を提供する ***metro*** オブジェクトを作成した．***metro*** オブジェクトは主に次の 2 つをサポートする．

- 小節の頭出し
- 拍数・小節数と秒数の相互変換

metro.barsToMs(bar, bpm) は，与えられた ***bpm*** を元に ***bar*** 小節の長さをミリ秒に変換する．***metro.BPMToMsbeat(bpm)*** は，与えられた ***bpm*** を元に一拍あたりの長さをミリ秒に変換する．

metro.waitToNextHead(bar) は，小節の頭出しをサポートする．Promise オブジェクトを返却する関数で，***bar*** 小節後に返却した Promise が解決される．解決された Promise の引数には現在の BPM 情報と，解決されたタイミングのミリ秒時刻が渡される．これらの情報を元にユーザはプログラミングを行う．例として，再生中の小節の 1 小節後の頭に DeckB の楽曲再生を開始した後，さらに 2 小節後の頭で DeckA の楽曲再生をストップするプログラムを示す．

```
1 metro.waitToNextHead(0)
2   .then((t, o) => {
3     deckb.play();
4     return metro.waitToNextHead(1);
5   });
6   .then((t, o) => {
7     decka.stop();
8   });
```

4.4.3 パラメタ制御オブジェクト

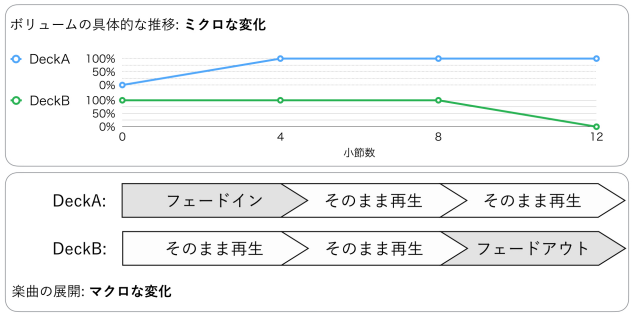

図 5 DJ プレイをプログラムする対象の分析: DeckB から DeckA へのフェードイン・フェードアウトでのミックスの場合

DJ プレイは曲の展開に合わせて曲に変化を付け，ミックスすることが多い．多くの楽曲は数小節ごとに展開するので，イコライザやボリュームなどの変化は数小節ごとの単位で入れ替わる．ここで，プログラミングしたい対象を大きく 2 つに分類できる（図 5).

- 細かなパラメタの操作(ミクロな変化，図 5 上部): 4 小節かけて線形にボリュームを大きくして曲をフェードインさせる．
- 幅を持った展開の流れ(マクロな変化，図 5 下部): 4 小節かけて次の曲(DeckA)を混ぜ，4 小節待機，4 小節で元の曲(DeckB)をフェードアウトさせる．

図 5 で例に取っているような，ある曲からある曲へフェードイン・フェードアウトで楽曲を繋げる DJ ミックスをプログラムすることを考える．どのようにボリュームを変化させるのかといった細かなパラメタの操作は，f(x) の関

数として表現すればよい．一方で，DeckA を混ぜた後 4 小節後はどちらも待機，その 4 小節後に DeckB をフェードアウトするといった，おおまかな展開を表現するようなプログラムを素直に記述しようとすると，自分で時間を管理しながら条件分岐等で展開を振り分ける必要がある．

そこで DJ ミックスの展開は数小節単位で起こることに着目し，時間管理を極力容易にするための API を構築した．関数に対して小節の長さを紐付け，関数を合成して提供するオブジェクトを作成した．

impl()メソッドはパラメタ制御オブジェクトを生成する．この戻り値に対してメソッドチェーンで関数をつなげていくことで展開を表現する．戻り値に対して呼ぶことの出来る関数は２つで，***phase(bar, fn)***と ***end()***がある．

phase(bar, fn)は与えられた関数 ***fn*** を ***bar*** 小節の展開として保存する．関数 ***fn*** は引数にその小節間の推移を 0~1 で示す ***t*** を取る．***end()***は与えられた小節数と関数の組み合わせを元に，0~1 を引数に取る関数を返却する．例として，図 5 で取り上げられたミックスの値推移を表現するプログラムを示す．

```
var transitionA = impl()
  .phase(4, t => { return t })
  .phase(8, t => { return 1 })
  .end();
var transitionB = impl()
  .phase(8, t => { return 1 })
  .phase(4, t => { return 1 - t })
  .end();
```

4.4.4 ミックスパターン生成支援関数群

本研究は DJ プレイをプログラマブルにすることで新たな DJ プレイの表現を探るものである．既存の DJ プレイなどから，多用するようなパターンを生成しやすくする関数を用意した．

tk2(v, t)は，***2v***[Hz]の矩形波を出力する関数である．***t*** は 0~1 で値をとる．パラメタ制御オブジェクトと組み合わせて使われることを意識しており，関数名の頭に r を付けると値が反転した関数が手に入る．この他にも三角波を出力する ***tri(v, t)***関数などが実装されている．

4.5 サンプルプログラム

これまでに紹介した API を駆使した例として，2 曲を交互に素早く入れ替えながらミックスするプログラムを示す．音量フェーダーを正確に操作しつつエフェクトの操作も行っており，人間一人では出来ない演出が可能となっている．

```
//プレイ時にどちらのデッキがa, bか迷わないための工夫
var before = decka;
var next = deckb;

// 次の曲のフェーダー推移
var nextFn = impl()
  .phase(4, (t, f) => { return t })
  .phase(1, (t, f) => { return rtk2(4, t) })
  .phase(1, (t, f) => { return rtk2(8, t) })
  .phase(1, (t, f) => { return rtk2(16, t) })
  .phase(4, (t, f) => { return 1; })
  .end();

// 現在の曲のフェーダー推移
var beforeFn = impl()
  .phase(4, (t, f) => { return 1 })
  .phase(1, (t, f) => { return tk2(4, t) })
  .phase(1, (t, f) => { return tk2(4, t) })
  .phase(1, (t, f) => { return tk2(8, t) })
  .phase(1, (t, f) => { return tk2(16, t) })
  .phase(4, (t, f) => { return tk2(32, t)*(1 - t); })
  .end();

// 次の小節頭からミックスをスタート
metro.waitToNextHead(0)
  .then(o => {
    // 12小節が何ミリ秒かを計算
    var barms = metro.barsToMs(12, o.bpm);

    // 次の曲を再生開始
    next.play();
    var id = setInterval(() => {
      // 12小節のうちの割合を計算
      var t = (Date.now() - o.time) / barms;

      // フェーダー操作部分
      before.volumes.setFader(beforeFn(t) * 127);
      next.volumes.setFader(nextFn(t) * 127);

      // エフェクト設定
      if(t > 2/4) {
        before.fx.a.on();
        before.fx.b.on();
        if(t < 3/4 ) {
          next.fx.a.on();
          next.fx.b.on();
        } else {
          next.fx.a.off();
          next.fx.b.off();
        }
      }
    }, 10);

    setTimeout(() => {
      clearInterval(id);
    }, barms);
  });
```

5. 実装

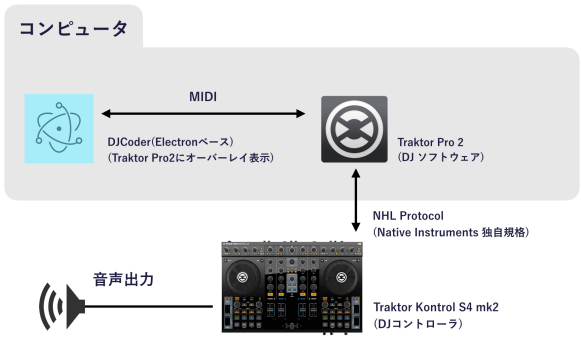

図 6 システム構成図

本章では，これまでに述べたインタラクションデザインを実現するための実装手法を簡単に紹介する．システム構成を図 6 に示す．システムのソフトウェア部分は Native Instruments 製の DJ システム Traktor Pro 2[a]に Electron[b]べ

[a] “Traktor Pro 2”. http://www.native-instruments.com/products/traktor/dj-software/traktor-pro-2/, (参照 2016-07-02).

[b] “Electron - Build cross platform desktop apps with JavaScript, HTML, and CSS.”. http://electron.atom.io/, (参照 2016-07-05).

ースのアプリケーション DJCoder をオーバーレイする形で構成している．Traktor と DJCoder は MIDI で通信を行っている．

現状では曲とプログラムの紐付けを行うことができないため，任意のディレクトリにまとめて保存し，逐一ユーザがコピー・ペーストをしてエディタに貼り付ける必要がある．

6. 議論

6.1 本手法による制約と拡張性

本システムは DJ システムを外部から MIDI でコントロールし，UI をオーバーレイするという手法で実装されているため，手法上の限界がある．

まず，使用している DJ システムである Traktor Pro 2 の限界に依存する．例として，5 デッキ以上，すなわち 5 曲以上を扱うようなミックスが行えないことや，逆再生が行えないことが挙げられる．

また，DJ システムが外部からコントロールするための手段を提供していない場合も機能が制限される．例えばプログラム側から選曲を行うことや，楽曲情報を取得して経過時間に基づいたプログラムを記述することなどは行えない．楽曲の信号を取得して，信号処理を行うようなプログラムも不可能である．

使用している DJ システムにオーバーレイできるようユーザインタフェースを設計しているが，その配置は数値で決め打ちとなっている．そのため，DJ システムのユーザインタフェースが変更された場合には，提案手法側の実装も変更する必要がある．

一方で，実装が済んでいないがための制約も存在する．例えば，Traktor Pro 2 は 4 つのデッキが容易されているが，現在の実装ではそのうち 2 デッキのみ実装が済んでおり，3 デッキ以上のミックスは行えない．また，ループ機能やジャンプ機能を利用したミックス，途中でテンポが変化するようなミックスも行えない．

すなわち，Traktor Pro 2 の機能で，外部から MIDI で操作可能なものであればプログラムから利用可能なように拡張できる．また，他の DJ システムであっても，DJCoderEditor の出力する MIDI 信号に対応するよう設定を記述すれば，その DJ システムを利用した DJ プレイを自動化することができる．

6.2 DJ プレイ中のプログラミング

DJ プレイ中の楽曲変更にあわせてプログラミングを行おうとしても，プログラミングに使える時間は次の曲をミックスするまでの時間で，非常に短いことが多い．現実的な変更は小節数などのパラメタ調整や，展開の入れ替え程度であろう．そうした場合は直接プログラムを書き換えるのではなく，当日変更しそうな部分を GUI で抜き出しておけるような仕組みが必要であると考えられる．

Kato らの TextAlive [9]では，プログラムのコメント中で任意のパラメタ GUI を宣言することができ，プログラムを読み書きせずとも GUI でパラメタの調整を行える．

こうした GUI のパラメタ調整と楽曲の解析情報を組み合わせることで，DJ プレイ中の楽曲変更にも耐えることが可能になるのではないかと考えている．

6.3 その他のプログラミング手法の検討

DJ プレイ中は「8 小節後にボリュームを上げて…」というよりは，展開を聞きながら次の動作を考えて DJ ミックスを行うことが多い．一方で今回のプログラミング手法は，数小節ごとのフェーズに分けてパラメタの移動を定義し，関数を生成して利用する手法を取った．実際の DJ プレイとは捉え方が変わってくるため，プログラムと頭のなかでのイメージを整えていく作業が難しい．プログラミング手法を改善することで，本番でのミスの軽減や，開発サイクルの高速化を実現できるはずである．

6.3.1 Programming by Example(PbE)

エンドユーザプログラミングの手法として，ユーザの動作を記録しプログラムを作成する Programming by Example(PbE)がある．一旦ユーザが DJ プレイを行い，そのプレイのコントローラの動作をプログラムで記録しおおまかな展開をつくり，プログラムを編集していくような開発手法が適している可能性もある．

コントローラの動作を記録できれば，ある楽曲に対するつなぎを収集することができる．楽曲情報と照らし合わせることで，つなぎのパターンを自動生成する等のアプリケーションを実装することも可能だと考える．

6.3.2 Live Coding

現在は一度プログラムを実行させてしまうと，その後修正を行うことが出来ない．「あともう二小節くらい後に展開を切り替えたい」と途中で感じたとしても，一旦プログラムを停止させ，書き直し，楽曲位置を戻し，もう一度プログラムを実行する必要がある．プログラムを変更した際に，実行しているプログラムも変更することができれば，さらに柔軟な開発やパフォーマンスが行える．

プログラムを実行中にプログラムを書き換えることで，実行中のプログラムを変更する手法を Live Programming と呼ぶ．また，Live Programming を利用して即興で音楽制作・パフォーマンスを行う表現手法を Live Coding と呼ぶ．SuperCollider[1]や ChucK[2]なども Live Coding ツールの 1 つであるが，前途した GUI パラメタ調整機能や，楽曲の解析情報と組み合わせることで，より柔軟なプログラミング環境が構築できるのではないかと考えている．

6.4 DJCoder 専用ハードウェアの検討

DJ 経験者への事前取材でも分かったように，PCDJ では満足の行くパフォーマンスが行えないため，CDJ で DJ プレイを行う DJ が存在する．Native Instruments 社の Traktor Kontrol S8[10]では，DJ コントローラの液晶パネルで選曲か

らミックスまでの一連のフローを行うことができ，PC に向かうこと無しに PCDJ を活用した DJ を行うことができる．

本システムも，コーディングやプログラムの実行などの動作がコンピュータに集約されており，パフォーマンスとして DJ を捉えた場合に支障が出る可能性は十分考えられる．MIDI パッドコントローラにプログラムを事前に割り当てておき，同時押し等で意味付けを行いながら実行するような，新たなプログラム実行手法が構築出来ないかと考えている．こうしたプログラム実行手法が構築できれば，既存の DJ ブースに DJCoder が溶け込ませた上でパフォーマンスを行えるようになる．

7. まとめ

本論文では，コンピュータと人間が共同でミックスを行う DJ プレイ手法・およびそのためのプログラミング環境「DJCoder」を実装および提案を行った．研究上の貢献は，1)事前に記述したプログラムを実行することで，人間には難しい DJ ミックスが可能になった点と，2)DJ に特化したタイミング制御 API を提供することで，ユーザが DJ のプログラミングを行いやすくした点である．今後は DJCoder によるパフォーマンスやユーザテストを行い，プログラミングパターンの発見やインタフェースの改善を行っていきたい．

謝辞 藤原 裕也さん，佐々木 佳祐さん，伊藤いずみさんに 3 章での取材に協力していただいた．本研究の一部は JST CREST の支援を受けた．

8. 参考文献

[1] McCartney, J. (1996). SuperCollider, a new Real Time synthesis language. In*Proceedings of the 1996 International Computer Music Conference* (pp. 257-258). The International Computer Music Association.
[2] Wang, Ge, and Perry R. Cook. "ChucK: A concurrent, on-the-fly audio programming language." Proceedings of International Computer Music Conference. 2003.
[3] “Max is a visual programming language for media. | Cycling '74”. https://cycling74.com/products/max/, (参照 2016-07-02).
[4] McLean, Alex. "Making programming languages to dance to: live coding with tidal." Proceedings of the 2nd ACM SIGPLAN international workshop on Functional art, music, modeling & design. ACM, 2014.
[5] Hirai, Tatsunori, Hironori Doi, and Shigeo Morishima. "MusicMixer: Computer-Aided DJ System based on an Automatic Song Mixing." Proceedings of the 12th Advances in Computer Entertainment Technology Conference. 2015.
[6] “Algoriddim - djay for Mac, iPad, iPhone, Android”. https://www.algoriddim.com/, (参照 2016-07-02).
[7] Fujima, Jun, et al. "Clip, connect, clone: combining application elements to build custom interfaces for information access." Proceedings of the 17th annual ACM symposium on User interface software and technology. ACM, 2004.
[8] Yeh, Tom, Tsung-Hsiang Chang, and Robert C. Miller. "Sikuli: using GUI screenshots for search and automation." *Proceedings of the 22nd annual ACM symposium on User interface software and technology*. ACM, 2009.
[9] Kato Jun, Tomoyasu Nakano, and Masataka Goto. "TextAlive: Integrated Design Environment for Kinetic Typography." *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 2015.
[10] ”Traktor : DJ Controllers : Traktor Kontrol S8 | Products". http://www.native-instruments.com/products/traktor/dj-controllers/traktor-kontrol-s8/, (参照 2016-07-02).