演者の手の動きを取り入れた ライブコーディングによる映像表現の試み

中田 航平[1]　平林 真実[1]　小林 孝浩[1]

概要：ライブコーディングは音楽表現に用いられることが一般的だが、近年映像表現に用いられることが増えている。現状のライブコーディングによって提示されるものは操作される画面の情報のみであり、演者自身の動きまでは提示されない。本稿では、LeapMotion を用いて演者の手の動きを取得し演出に取り入れたライブコーディングによる映像表現を試みた。本稿で試験的に行ったパフォーマンスの映像構成と演出では、演者の手の動きとの関連をすぐに認識するまでには至らず、手を表現するオブジェクトの多様化を始めとした表現の工夫等、今後の改善が望まれる。

1. はじめに

ライブコーディングは、即興でプログラムを記述することで音楽や映像を生成するパフォーマンスである。ライブエレクトロニクスミュージックを起源としているライブコーディングは、現代においても音楽表現に用いられることが一般的である。その一方で、近年の GPU の性能向上などによりリアルタイムでの映像生成が可能となったことから、ライブコーディングによる映像表現も増えてきている。

ライブコーディングを推進するコミュニティである TOPLAP の打ち出すマニフェストに、「Show us your screen」という言葉がある [1]。これは、ライブでのコンピュータ音楽に見られる、演者が何をしているのかわからないという問題を解決するパフォーマンスであるライブコーディングにおいて、必ず満たすべき要件である。観賞者は演者が操作する画面を見ることができ、そうすることによりコンピュータ音楽においても他のパフォーマンスと同様に演者の行為を楽しむことができる。しかし、ここで演者から提示されるのはあくまでコンピュータへの入力結果であり、実際の演者の動きまでは提示されない。他のパフォーマンスにおいて演者の動きを見て楽しむことがあるように、ライブコーディングにおいても演者の動きを楽しめる表現が可能ではないかと考えた。

本稿では、ライブコーディングによる映像生成の新しい表現の試みとして、生成する映像に演者の手の動きを取り入れる。

[1] 情報科学芸術大学院大学 (IAMAS)

2. 関連研究

2.1 ライブコーディングによる映像生成

前述の通り、ライブコーディングは音楽表現を中心に発達してきた。現在 TOPLAP で用いられるチャットツール [2] で最も多くのユーザが参加している TidalCycles[3] を始めとし、SonicPi や SuperCollider など、古くから音楽生成のためのライブコーディング環境が開発されてきた。

一方、hexler によって開発された KodeLife[4] など、映像生成のためのライブコーディング環境も開発されている。Unity エバンジェリストの高橋啓次郎は、Unity と KodeLife を組み合わせたライブコーディングを行っている [5]。KodeLife で生成した映像を背景に、Unity で中心となるオブジェクトやパーティクルなどのエフェクトを描画し、ブラーなどポストエフェクトをかけている。

このようなライブコーディングによる映像生成は、コンピュータで映像を生成する特性上、プリミティブ図形など CG 要素の強いものであることが多い。本稿では、ライブコーディングにおいてもリアルタイムな現実の情報を重視した映像表現を模索する。

3. 作品「Physicality of Live Coding」

3.1 概要

本作は、演者の手の動きをライブコーディングによって生成される映像に取り入れることを試みたパフォーマンス作品である。演者は通常のライブコーディングと同様に、即興でプログラムを記述し映像を生成する。その際、演者の手元に設置されたデバイスから手元の情報を取得し、映像に反映させる。

© 2018 Information Processing Society of Japan

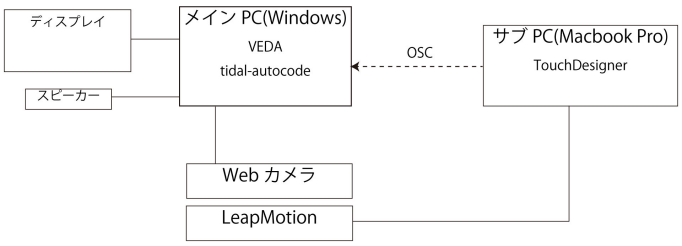

図 1　システム構成

3.2　実装、システム

本作のシステムについて述べる。**図 1** は、本作のシステム構成図である。本作では、Atom 上で GLSL ライブコーディングを行うためのパッケージである VEDA[6] を用いて映像を生成した。VEDA は Atom のパッケージであるため、画面構成、操作方法は通常 Atom を利用する場合と変わらない。VEDA では、シェーダー言語である GLSL を記述し実行することで、記述されたプログラムによって生成された映像を Atom エディタの背景に描画する。VEDA は音声や MIDI 等の各種外部入力に対応しており、本作では OSC と Web カメラの入力を受け、プログラムに反映させた。

手の情報の取得には Leap Motion と Web カメラを用いた。Web カメラによって手元の画像情報を、LeapMotion によって手の部位の位置情報を取得した。VEDA は直接 LeapMotion で得られるデータを扱う方法を持たない。よって、演者が使用する PC とは別の PC を用いて LeapMotion のセンサデータの取得を行い、得られたデータを OSC を用いて VEDA に送信した。

3.3　実演

2018 年 8 月 4 日に行われた情報科学芸術大学院大学のオープンハウスにて、15 分間のライブコーディングを計 3 回行った。**図 2** は、その際の実演の様子である。時間の設定は、映像のためのライブコーディングの特性上、短すぎるとプログラムを記述する映像に反映させる前にパフォーマンスが終了してしまうことを考慮した。回数は実演の場の特性を踏まえ多くの人に観賞してもらうための配慮である。

本作は映像表現が主体となるが、音声を全く使用せずライブコーディングを行うことは、観賞者の関心を維持することが難しく現実的ではない。しかし、映像表現を行う演者には可能な限り音楽生成には意識をとらせず、映像の生成に注力させることが、演者の能力を引き出す上で重要である。そこで今回は、音楽表現のためのライブコーディングツールである TidalCycles を自動で編集し実行する Atom パッケージである tidal-autocode[7] を用いた。これによって、演者は音楽表現に思考を割くことなく、音楽からもライブコーディングの特徴を感じてもらうことができる。

表 1　本作でのシーケンス

開始時間	内容
0:00	メインオブジェクト (tidal-autocode を起動)
0:05	背景 (Web カメラ画像を背景に設定)
0:10	ポストエフェクト

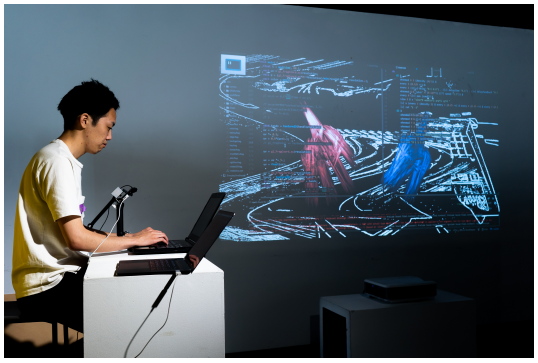

図 2　実演の様子

映像の構成は、kezzardrix が述べているオブジェクトコンポジション [8] を参考にした。オブジェクトコンポジションは、映像をいくつかの要素で成り立つものととらえ、それらを 3D 空間上でどう配置するかを考えながら独立して編集する映像構成の考え方である。本作では手の部位の位置情報から生成したオブジェクトをメインオブジェクトとし、Web カメラによって得られた画像情報を背景として映像を生成した。あ

本作では、**表 1** の手順でライブコーディングを行った。時間経過と共に要素が増えていくような時間構成となっている。

事前準備として、OSC で手の情報を取得できる機能までは事前に作成した。その他にもノイズやガウシアンブラーなど、比較的使用する可能性は高いが無意味に記述量が多いものは、ユーティリティとして事前に作成し、すぐに利用できるようにした。その他については、パフォーマンスの中でリアルタイムに行った。

4.　考察

4.1　構成

前述の通り、本作ではメインオブジェクトに手の部位の位置情報から生成した手のオブジェクトを、背景に Web カメラで取得した画像情報を配置し、それらをライブコーディングによって編集した。この映像を構成する要素や時間的な構成について、実演によって得た結果を考察する。

© 2018 Information Processing Society of Japan

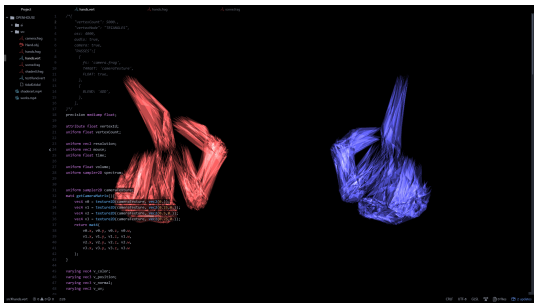

図 3　メインオブジェクト

4.1.1　メインオブジェクト

本作では直接的な手のモデルを用いず、その形状に近いオブジェクトを生成したが、抽象度が高く、実際の手の動きとの関連がわかりにくかった。パフォーマンスの初期段階ではより実際の手との関連がわかるようなオブジェクト、例えばリアルな手のオブジェクトを用いることでわかりやすさが改善される可能性があると考える。しかし、ライブコーディングの演出として、今回のような抽象的なオブジェクトの方が直接的な表現よりも面白いという意見もあった。

タイピングをする際の指の動きの絶対量は想定よりも小さく、位置情報をそのまま使用すると指の動きがわかりにくかった。これに対しては、位置情報から速度を計算しそれに応じてエフェクトを強調するなど、動いていない部位と差別化をすることでより演者自身の動きがわかりやすくなると考えられる。

4.1.2　背景

背景に Web カメラの映像を用いたことで、用いない場合と比較して演者の動きと映像の関連が深まったように感じた。キーボードや手元の周囲の情報も含んでいることと、演者の視覚により近い情報であることが要因であると考えられる。

背景にパラメータとして手の情報を反映させるのは、当初の想定以上に難しかった。今回用意したデータの取得方法では、手の情報を取得するまでの記述量が比較的多く、パラメータとして利用するには可読性や再利用の際の手間がかかりリアルタイムで記述するライブコーディングでは扱いが難しかった。

最終的に複数のエフェクトを背景にかけたが、今回の演出では演出としての面白さと演者の手元の関連のわかりやすさは相反することがわかった。パフォーマンスを途中から観賞し始めた人には、背景が手元の情報を用いて生成されていることに気づけないと考えられる。

4.2　ライブコーディングによる映像表現

本稿の趣旨以外にも、ライブコーディングによる映像表現における課題が見えた。

4.1.2 で述べたように、ライブコーディングにおいて記述量は大きな課題となる。ライブコーディングにおいても他のパフォーマンスと同じように観客を飽きさせないためのシーンの展開が必要となる。映像表現のためのライブコーディングでは、音楽表現のためのものと比べて様々な効果を付与するために多くの記述を要する。記述量が多くなるほど展開は少なくなり、観客が飽きるリスクが高まる。そのため事前準備において、可能な限りパフォーマンス中の記述量を減らす工夫が必要となる。しかしその一方で、短縮されすぎたプログラムは視覚的に見にくく、アルゴリズムが理解しにくい問題も起こりうる。これについては、パフォーマンスを行う場を考慮し表現としてどこを重要視するか構成を考える必要がある。

また、記述中の展開の工夫として、タイプしたキーをフラグとして映像を生成することも考えられる。即興で記述するライブコーディングでは、どのキーをタイプするかはその場で初めてわかる。そのため、乱数的な要素としてキータイプを映像生成に用いることで、記述中にも映像の変化を楽しめるようにすることが可能になると考えられる。

5.　おわりに

本稿では、ライブコーディングによる映像表現の試みとして、演者の手の動きを映像に取り入れた。今後の拡張として、メインオブジェクトとなる手のオブジェクトの多様化など表現としての工夫や、事前準備における記述量の短縮化やキータイプによる演出などのシステム面での工夫が考えられる。本稿がライブコーディングによる映像表現の一助となると幸いである。

参考文献

[1] TOPLAP: ManifestoDraft - Toplap(online), 入手先 ⟨https://toplap.org/wiki/ManifestoDraft⟩ (2018/08/06).

[2] TOPLAP: () LURK(online), 入手先 ⟨https://talk.lurk.org/home⟩ (2018/08/06).

[3] tidalcycles: Tidal - Home(online), 入手先 ⟨https://tidalcycles.org/⟩ (2018/08/06).

[4] hexler: hexler.net — KodeLife(online), 入手先 ⟨https://hexler.net/software/kodelife⟩ (2018/08/06).

[5] keijiro takahashi: Live coding experiment (KodeLife & Unity)(online), 入手先 ⟨https://vimeo.com/213872212⟩ (2018/08/07)

[6] amagitakayoshi: VEDA - VJ app for Atom(online), 入手先 ⟨https://veda.gl/⟩ (2018/08/06).

[7] kindohm: tidal-autocode(online), 入手先 ⟨https://atom.io/packages/tidal-autocode⟩ (2018/08/07).

[8] 神田竜: プログラミングを用いた映像制作のコツ～コンポジションによる絵作り～(online), 入手先 ⟨https://www.ameet.jp/digital-imaging/2094/2/⟩ (2018/08/06).

© 2018 Information Processing Society of Japan