Vol.2014-MUS-104 No.15
2014/8/26

TextAlive: 音楽に同期した歌詞アニメーションの Kinetic Typography 制作環境

加藤淳 [1,a)]　中野倫靖 [1,b)]　後藤真孝 [1,c)]

本稿では、歌詞を歌声と同期してアニメーションさせる Kinetic Typography と呼ばれる動画表現の制作環境 TextAlive を提案する。既存の制作ツールでは、歌詞と歌声の同期を手作業で取り、文字や単語、複数単語から成るフレーズに対して個別に望みの動きを設計する必要があった。その際は、動きを規定するアルゴリズムのパラメタを、スライダーなどの汎用 GUI で調整して試行錯誤を重ねていた。一方、本制作環境では、歌詞と音楽の時間的対応付けを自動で推定し、動きのアルゴリズムに対する初期パラメタを自動生成する。さらに、動きのアルゴリズムを編集できるコードエディタを備え、プログラマがパラメタ調整に適した専用 GUI を容易に提供できるフレームワークを提供する。これにより、TextAlive のユーザは Kinetic Typography を一から作る必要がなくなり、初めに時間合わせなどを行う手間をかけずに済む。また、歌詞の動きをインタラクティブかつグラフィカルに設計できるようになる。

1. はじめに

音楽の楽しみ方はメディア技術の発展に伴って変遷してきた。音楽のみを再生する機器は蓄音機以来数多く存在したが、テレビが普及して PV（Promotion Video）が制作されるようになり、音楽に視覚的な動画情報を加えて楽しめるようになった。また、カラオケはもともと歌声を除いた伴奏のみを流す装置だった。しかし、メロディーと同期して歌詞を表示できるビデオカラオケが登場し、歌詞カードを見るより容易に歌唱タイミングが分かるようになった。

動画メディアの文脈では、文字情報の持つメッセージをより印象付けるため、タイトルやテロップをアニメーション表示させる Kinetic Typography と呼ばれる表現手法が一般的である。これは印刷物上で文字を綺麗に配置する Typography の動画版であり、典型例としてテレビ番組のタイトルやテロップがアニメーションする動画が挙げられる。歌声は歌詞という文字情報を含むため、歌声と同期する動画制作において Kinetic Typography が利用されるのはごく自然な流れである（**図 1b**）。実際に、歌詞が音楽に合わせて派手にアニメーションする PV やビデオカラオケの動画が数多く制作されている。

しかしながら、文字情報を歌詞の発声タイミングと同期して思い通りにアニメーションさせることは容易ではない。その際は、動画を一から制作できる Adobe After Effects [1] などの汎用動画制作ツールを用いる必要がある。汎用ツールでは、各文字の表示位置や大きさ、フォントのスタイルなど様々なパラメタを、基本的に手作業で時系列に沿って指定していく膨大な手間がかかる。一部の音楽プレイヤーでは、歌詞と発声タイミングに関する情報を含むファイルを読み込み、音楽と同期して表示させるシンプルなカラオケ表示（**図 1a**）を行うことが可能だが、動画制作者の意図を含めることはできない。

1 産業技術総合研究所
National Institute of Advanced Industrial Science and Technology (AIST)
a) jun.kato [at] aist.go.jp
b) t.nakano [at] aist.go.jp
c) m.goto [at] aist.go.jp

このように、歌声と同期して歌詞がアニメーションする動画は音楽の楽しみを広げてくれる一方で、制作過程は現状、膨大な手作業で支えられている。本研究では、音楽と同期した歌詞アニメーションの Kinetic Typography に特化した制作環境 TextAlive を開発し、当該動画の作成・編集を支援する。本稿では、まず 2 章で歌詞アニメーションの Kinetic Typography を制作するうえで困難なポイントを整理する。次に、3 章で 1) 音楽と歌詞の時間的対応付けを半自動で行って制作にかかる手間を削減する手法と 2) 歌詞アニメーションに特化した直感的な動画編集を可能にする手法を提案し、これらを統合したシステム TextAlive について紹介する（**図 2**）。続く 4 章では実装について述べ、5 章で一般的な動画制作、音声と同期するマルチメディアの編集などに関する既存手法と比較して本研究の新規性について述べる。さらに、6 章で将来展望について議論したのち、結論を述べる。

2. 歌詞アニメーションの難しさ

本章では、一般的な動画制作と比べて歌詞アニメーションの Kinetic Typography 制作がとくに難しい理由について、

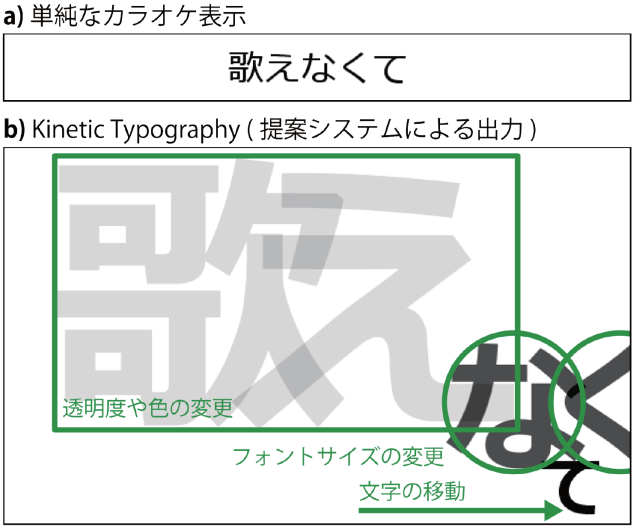

図 1　歌詞表示の方法

アニメーションの設計要素である時間（2.1）および空間的配置（2.2）のそれぞれの面から分析する。

2.1 歌声と文字情報の高度な同期

歌詞は自由詩であり、言語を問わず、文字と、その集まりである単語、さらに単語が複数集まったフレーズ（必ずしも主語述語などが揃った文にはならないためこう呼ぶ）の集合となっている。以降、このような文字情報のまとまりを文字ユニットと呼ぶ。歌詞をアニメーションさせた Kinetic Typography 動画では、個々の文字ユニットに対して画面上で最も目立って見える瞬間が効果的に設定されている。動画制作者の意図に応じて、一文字一文字が順番に表示され、発声終了時にフレーズ全体が最も目立つ場合もあれば、発声直前にフレーズが丸ごと表示され、最も目立つ場合もある。

このようなアニメーションを制作するためには、一般的な制作ツールでは全てのタイミングを手作業で合わせる必要があり、非常に手間がかかる。歌詞が歌声と単純に同期して表示されるカラオケの表示方法は Kinetic Typography の最も単純な例であり、実際の動画ではより高度な同期が行われている場合が多い。

2.2 空間的な意味を持つパラメタ調整

歌詞のアニメーション動画はすでに数多く制作されており、様々な典型的パターンがある。このような文字のアニメーションを Adobe After Effects [1]のような汎用の動画制作ツールで作成するためには、キーフレームにおける位置を順番に指定していくなど、低レベルなパラメタ指定を積み重ねていく工程が必要となる。すなわち、ユーザは、最終的な動きを頭の中に思い描きながら個別具体的なパラメタ調整を行う必要がある。このようなコマンド（パラメタ調整）と実行結果（生成されるアニメーション）の間の溝は Gulf of execution and evaluation [2]と呼ばれ、適切なインタフェースによる支援が必要とされる。

歌詞のアニメーションを典型的なパターンに当てはめると、実際に指定する必要があるのは歌詞の初期位置や、歌詞がアニメーションする向きなど、空間的な意味を持ついくつかのパラメタだけである場合が多い。しかし、各パターンに固有のパラメタをグラフィカルに指定できるインタフェースは汎用ツールでは提供されない。

3. TextAlive

本章では、前章で分析した問題を解決するための手法を提案し、これらを実装したシステム TextAlive を紹介する（**図 2**）。TextAlive は、制作者のスキルに応じて 2 通りの使い方ができるように設計されている。特定の前提知識を持たないユーザは、GUI で音楽に同期した歌詞アニメーションの Kinetic Typography 動画をインタラクティブかつグラフィカルに制作できる。さらに、プログラミングができるユーザは、歌詞の文字列をアニメーションさせるための動きのアルゴリズムを、文字ベースのプログラムとして実装、編集できる。プログラミングは必須知識ではないが、After Effects のような汎用の動画制作ツールでも複雑な動きの設計はスクリプト言語で行うほうが効率的であり、動画制作で必要とされることがあるスキルである。TextAlive で作成した Kinetic Typography のアニメーション事例は、研究プロジェクトの Web サイト[*1]で確認できる。

3.1 音楽情報処理を用いたアニメーションの自動生成

本節では、与えられた音楽ファイルと歌詞ファイルに対して歌詞アニメーションの初版を自動生成する手法を提案する。本手法により、ユーザは一から歌詞のアニメーションを作成する必要がなくなり、自動生成されたものをベースに編集して自分好みのアニメーションに近づけていくことが可能となる。

ユーザはまず、歌声が含まれる音楽ファイルと歌詞の文字列が書かれた歌詞ファイルを選択する。この後、システムが音楽と歌詞の時間的対応付けを推定[3]する。これにより、歌詞中の各文字、単語、フレーズの発声タイミングが仮に定まる。さらに、システムが楽曲のサビ区間を検出[4]し、これらの情報を用いてデフォルトのアニメーションを自動生成する。

発声タイミングの推定結果が誤っている場合、ユーザは、短い区間に絞って音楽と歌詞の時間的対応付けを推定させ直すか、タイムライン上で GUI による手作業により発声タイミングを編集できる（**図 3**）。発声タイミングが更新されると、アニメーションのタイミングも更新される。なお、修正されたタイミング情報は自動的に保存され、次に同じ組み合わせの音楽と歌詞を読み込んだ際に利用される。

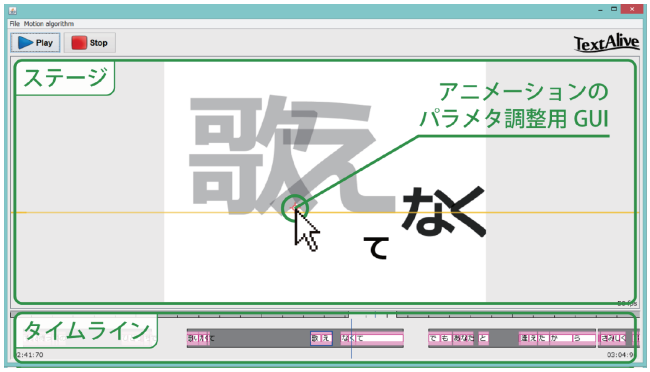

図 2 TextAlive のメイン画面

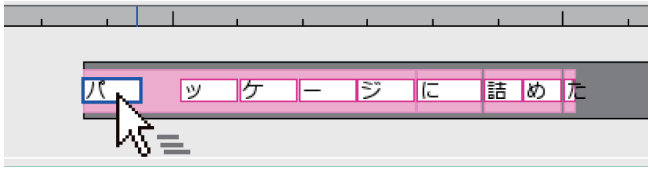

図 3 タイムライン上での発声タイミングの修正

[*1] TextAlive. http://staff.aist.go.jp/jun.kato/TextAlive/

3.2 インタラクティブなパラメタ調整用 GUI の提供支援

いったん歌詞アニメーションの初版が自動生成されると、ユーザは、ステージ上でアルゴリズムのパラメタを調整したり、別のアニメーションアルゴリズムを割り当てたりできる（**図 2**）。本節では、パラメタ調整用のユーザインタフェースをよりグラフィカルなものにするための手法を提案する。これにより、エンドユーザは目標とする動きに近づけるためのパラメタ調整をより直感的に行えることが期待できる。

既存の動画制作ツールでは、グラフィカルに調整できるパラメタがオブジェクトの初期位置などに限定されていた。したがって、実際にオブジェクトがどのような動きになるか視覚的に分からないことが多かった。GUI によるパラメタ調整が難しい要因として、動きのアルゴリズムには様々な種類があり、統一的なパラメタ調整用ユーザインタフェースが提供できないことが挙げられる。そこで TextAlive では、アルゴリズム実装者が一番アルゴリズムについてよく知っており、直感的なパラメタ調整用 GUI を実装できるという仮定を置く。そして、ユーザインタフェースの実装を容易にする手法を提案する。すなわち、TextAlive 上に動きのアルゴリズムを実装できるソースコードエディタを提供し、パラメタ調整用ユーザインタフェースをアルゴリズムと同時に実装できるようにする（**図 4**）。その際、ユーザインタフェースのひな形となる GUI ウィジェットの実装を Kinetic Typography 制作用に特化して用意し、単純なユーザインタフェースならソースコードにコメントを一行足すだけで提供できるようにする（**図 5**）。

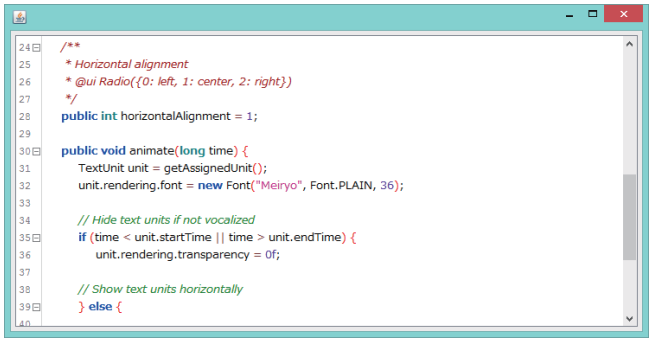

図 4 アニメーションのアルゴリズムとパラメタ調整用 GUI を実装するための Java ソースコードエディタ

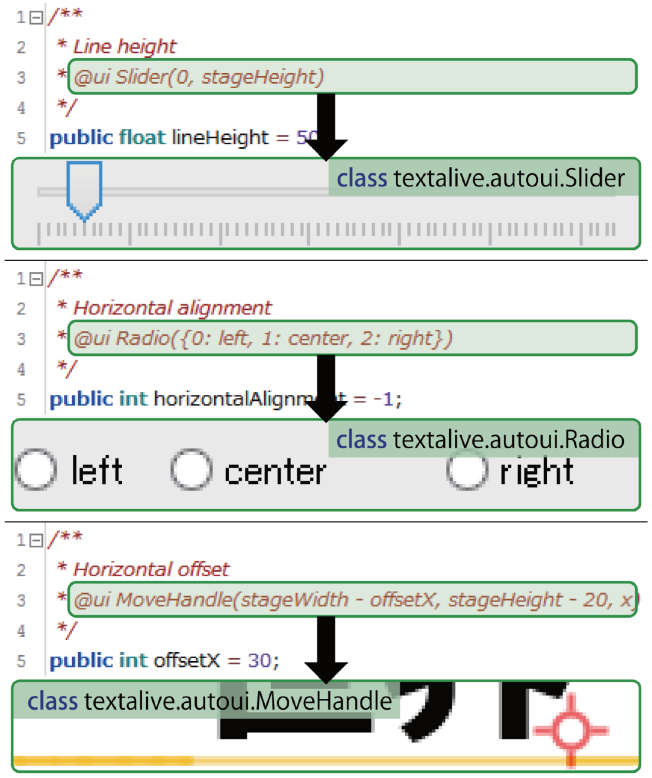

図 5 コメントを解析することでステージ上に生成されるパラメタ調整用 GUI

このように、ソースコード中の特定の部分を特別扱いし、アルゴリズム本体とは異なる評価を行う手法は、パラメタ調整を多用するプロトタイピング用途のプログラミング環境で提案されてきた（5.3 で詳述）。とくに静的コード解析によってコメント欄の内容を抽出し、パラメタ調整用 GUI にする手法は Juxtapose [5]が最初と考えられる。類似手法は統合開発環境 Unity[*2]にも採用され、C#のクラスで public 修飾子の付与されたフィールドに対して自動的にスライダーが表示されるようになっている。これらの既存手法ではスライダーのみ配置可能であったものを、本手法ではさまざまなユーザインタフェースの実装にも応用可能なように拡張した点に新規性がある。

4. 実装

本章では、前章の手法を実現した歌詞アニメーション制作環境 TextAlive の実装について述べる。

4.1 概要

TextAlive システムにおける処理の流れを**図 6** に示す。TextAlive は Java で実装されており、64bit 版 Windows 8.1 で動作確認を行っている。音楽ファイルは MP3 または MS WAV フォーマットに対応している。歌詞ファイルはテキストファイルの形式で与える。歌詞ファイルの内容は、フレーズごとに改行されたプレーンテキストである。単語は MeCab[*3]による形態素解析の結果として得られる。フレーズや単語のまとまりは、ユーザが後から再編集できる。なお、音楽ファイルが MP3 フォーマットの場合、メタ情報を埋め込む仕様である ID3v2.3.0[*4]に歌詞の埋め込み方法が明記されている。TextAlive は当該仕様に対応しているため、歌詞が MP3 ファイルに埋め込まれている場合、テキストファイルを別途用意する必要はない。

歌詞をアニメーションさせるためのアルゴリズムや、ア

[*2] Unity. http://unity3d.com/
[*3] MeCab. http://mecab.sourceforge.net/
[*4] ID3 v2.3.0. http://id3.org/id3v2.3.0

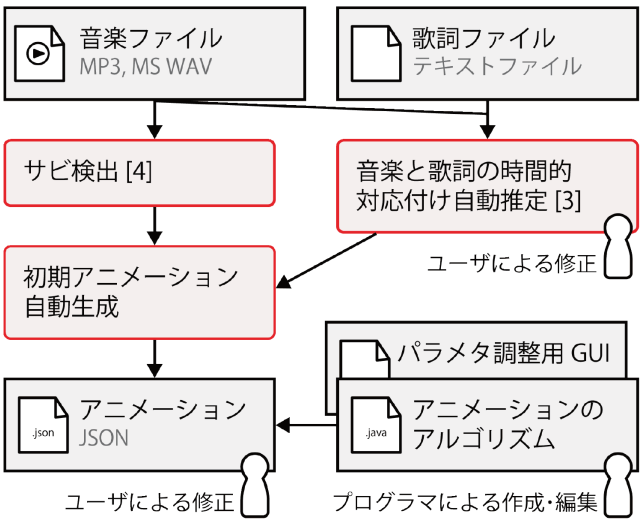

図 6 TextAlive システムにおける処理の流れ

ルゴリズムのパラメタ調整用 GUI は Java で記述される。これらのソースコードは、TextAlive の内蔵ソースコードエディタでいつでも作成・編集できる。また、編集内容はすぐにアニメーションや GUI に反映される。例えば、現在作成中のアニメーションで使われているアルゴリズムが編集されると、生成されるアニメーションも動的に更新される。

4.2 歌詞のアニメーション

歌詞のアニメーションでは、可読性を確保するため、単語単位で見ると文字が横方向や縦方向に順番を保って並んでいることが多い。また、単語が文字の相対的な位置関係を保ったまま動いたり、複数の単語がまとまってフレーズを形成し、フレーズごとに文字の重複を避けるような処理が行われたりする。このように、各文字ユニット（文字、単語、フレーズ）の位置や向きを決める処理は他階層のユニットとは独立しており、上位階層のユニットを基準にした相対座標系を用いるとシンプルに記述できる。

そこで、TextAlive では、ユーザが各文字ユニットに対して別々のアニメーションのアルゴリズムを割り当てられるようにしている。例えば**図 1** の例では、各単語に「文字の横書きを行いつつ発声開始に合わせて画面右端に到達するよう右方向にスライドするアルゴリズム」が、各フレーズに「画面縦方向にばらけさせ、文字が重なって読めなくなるのを防ぐ改行のためのアルゴリズム」が割り当てられている。それぞれ独立にアルゴリズムを切り替えられるため、例えば各文字に対して「回転するアルゴリズム」を追加したり、単語の移動する向きを左向きにしたりするなど、さまざまな組み合わせを試すことができる。

各瞬間における歌詞のレンダリング処理では、画面上の絶対座標系におけるフレーズの位置、フレーズ基準の相対座標系における単語の位置、単語基準の相対座標系における文字の位置が計算される。そして、各文字の表示に用いられるパラメタを画面上の絶対座標系に変換してゆき、画面内へ収まるものだけが描画される。

4.3 アニメーションの自動生成

デフォルトのアニメーションは、3.1 で述べたように、音楽と歌詞の時間的対応付けが確定した後、サビの開始、終了タイミングの情報を利用して生成される。ここで自動生成される内容はユーザがアニメーションを自分好みにしていくための土台となるため、シンプルかつ TextAlive の機能が伝わる分かりやすい具体例となっていたほうがよいと考えられる。そこで、現在の実装では次段落に示すようなヒューリスティックな方法でアニメーションのアルゴリズムを割り当てている。将来的には、この自動生成の方法もプログラマが作成、編集できるようにして、テンプレートとしてユーザが自由に選べるようにする予定である。

サビは楽曲が盛り上がる場面である。したがって、Kinetic Typography の演出上も、サビをそれ以外の区間より盛り上げたほうが曲調と合うと考えられる。まず、サビ以外については、各単語に対して「文字が右方向にスライドするアルゴリズム」が、各フレーズに対して「改行のためのアルゴリズム」が割り当てられるようになっている（先述した**図 1** の例）。一方、サビについては各文字に対して「画面を埋め尽くすように外周から右回りに配置されていくアルゴリズム」が割り当てられるようになっている。実際に画面を埋め尽くしそうになったら古い字から消えていく。また、配置されたてのときは文字色がオレンジだが、時間が経つと黒くなる。このように、画面が次々と鮮やかな色で埋まっていくため、サビ以外と比べて印象に残りやすい。

4.4 アニメーションとパラメタ調整用 GUI の Java 実装

アニメーションのアルゴリズムはそれぞれ一つの Java クラスとして定義されており、TextAlive 起動時にソースコードがコンパイルされて Java VM へ動的に読み込まれる。また、プログラマがアニメーション制作中にアルゴリズムを編集した際は、ソースコードが再度コンパイルされ、クラスが読み込み直される。通常の Java 用開発環境では、プログラム起動中にクラスを読み直す機能はホットスワップと呼ばれる。ただし、単なるホットスワップでは、今後割り当てられるアニメーションのアルゴリズムが新しくなるだけで、現在編集中のアニメーションで既に利用されているアルゴリズムは更新されない[*5]。そこで、4.5 で詳述する方法により、可能な限り古いアニメーションのパラメタを保ったままアルゴリズムを新しいものに差し替え、画面を更新する。こうして、プログラマがシームレスにプログラムの実行（アニメーション制作）とソースコードの編集（ア

[*5] ホットスワップはもともと Web サービスの実装などでよく使われる技術である。クラスをリロードした後も Java VM 上に古い実装のインスタンスが残るが、これは古いリクエストへの応答に使われているもので、新しいリクエストへの応答は新しい実装のインスタンスで行われる。この場合はサーバ応答の一貫性が保たれるため、むしろ望ましい動作と言える。

ルゴリズムの設計）を行き来できる Live Programming [6]を実現している。

また、アニメーションアルゴリズムの Java クラスが Java VM へ読み込まれるとき、同時に静的コード解析が行われ、パラメタ調整用 GUI を生成するための情報が収集される。具体的には、ANTLR Parser[*6]を用いてソースコードがパースされ、Java クラスの public フィールドがアニメーションの内容を調整するためのパラメタとして扱われる。パラメタ調整用 GUI は、当該 public フィールドに付与されたコメントの内容に応じて図 5 のように生成される。コメントは @ui に続けてパラメタ調整用 GUI の実装クラス名とパラメタの値域などを表すオプションを記述する仕様になっており、これによってウィジェットが適切な位置にレンダリングされ、適正な範囲でのパラメタ調整が行える。

パラメタ調整用 GUI として、スライダー（図 5 上）や選択式のラジオボックス（同中）アニメーション表示画面上で移動可能なハンドル（同下）といった代表的なウィジェットの実装は初めから提供されている。しかし、独自の GUI を実装する場合は、パラメタ調整用 GUI を表す基底 Java クラスを拡張して、新たなクラスを定義する必要がある。

4.5 アニメーションの保存と読み込み

TextAlive で作成、編集されたアニメーションを保存するには、歌詞を構成する文字ユニット（文字、単語、フレーズ）を時系列で並べ、それぞれについて音楽との時間的対応付け（発声開始と終了のタイミング）および割り当てられたアニメーションアルゴリズムの種類とパラメタを書きだせばよい。このうち、アニメーションアルゴリズムの種類はアルゴリズムを表す Java クラスの名前であり、パラメタはその public フィールドの名前と値のペアである。これらの情報を JSON オブジェクトとして表記すると、例えば **図 7** のようになる。アニメーションの読み込みは、この JSON オブジェクトをパースして、文字ユニットやアニメーションアルゴリズムのインスタンスを生成すればよい。

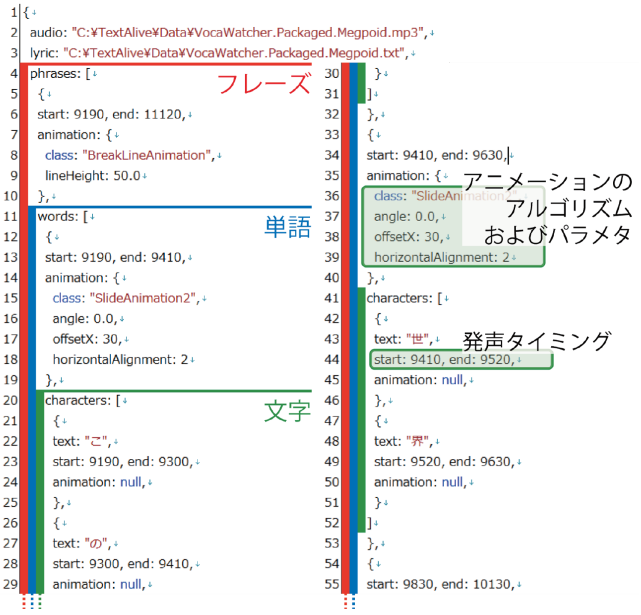

図 7 アニメーションを表す JSON ファイルの例（抜粋）

なお、いったん JSON として書き出したアニメーションを再度読み込む際に、アニメーションアルゴリズムの実装が更新されている可能性がある。そこで、例えば JSON 側に保存されているパラメタが Java クラスに当該フィールドがないために読み込めない、あるいは JSON 側で値が未定義のパラメタが Java クラスでは宣言されていることが起きうる。このミスマッチは、別のアニメーションの編集中にアルゴリズムの実装を変更した場合、またはアニメーション制作中に利用しているアルゴリズムの実装を変更した場合（4.4 で言及）に生じる。このようなケースでは、JSON 側にしかないパラメタを無視し、Java クラス側にしかないパラメタは初期値を代入することによって、可能な限り元々のパラメタを保持するように処理する。

5. 関連研究

5.1 Kinetic Typography の制作支援

Kinetic Typography は、文字情報を動画としてアニメーションさせ、視聴者の印象に残す表現手法である。実際に、静止した文字と比較して感情的な情報をより伝えやすいという実験結果[7]が報告されている。この表現手法を容易に実現するために、プログラミング言語[8]やフレームワーク[9]についてのプログラマ向けの研究の他、GUI で文字列に動きをつけられるエンドユーザ向けのユーザインタフェース[10]が提案されてきた。エンドユーザ向けツールの研究成果は Web 上で公開され、様々なメディアアートに応用されている[11]。文字列に対してテキストマイニングを行い、想起される感情を推定して適切な動きを自動選択する手法[12][13]も提案されている。汎用の動画制作ツールに Kinetic Typography 制作用インタフェースを追加するプラグイン[14]が発売されているが、テンプレートで用意された動きしか適用できない。

このように、Kinetic Typography に関する既存手法はいずれも文字のアニメーションのみを扱った研究や製品である。一方で本研究は歌詞がアニメーションする動画に着目し、音楽と同期する制約条件を利用して半自動的なユーザインタフェースを提供している。

5.2 アニメーション制作用インタフェース

汎用のアニメーション制作ツールとして、Adobe Flash や After Effects [1]などが市販されている。これらのツールでは、時間軸に沿ってキーフレームを指定し、キーフレームにおけるオブジェクトの状態を指定すると、間のフレームを滑らかに補間することにより、プログラミング不要でア

[*6] ANTLR. http://www.antlr.org/

ニメーションを制作できる。Microsoft PowerPoint のようなスライド作成ツールでも、スライド上のオブジェクトに簡単なアニメーション効果を指定できる。

研究では、ペンデバイスやタッチ入力によりオブジェクトが動く軌跡を描くインタラクション手法[15]が提案されている。また、軌跡に沿って色や拡大率などオブジェクトの見た目に関するパラメタを変化させることで、オブジェクトを動かしながら徐々に色や大きさを変えるアニメーションを制作できる手法[16]が提案されている。

上記の手法は、いずれも個々のオブジェクトの動きを手作業で指示する必要がある。そこで、水の流れや鳥の群れ、木から散る葉のように、多くのオブジェクトが同じような傾向を持って動くアニメーションをスケッチ入力で描くためのインタフェース[17]が提案されている。しかし、出現するオブジェクトの数などは細かく指定できない他、個々のオブジェクトに対して詳細なパラメタ調整ができない。

本研究では、個々の文字でなく単語やフレーズに対してアニメーションを割り当てることで、複数オブジェクトの動きを一気に指定できる。また、アニメーションのパラメタ調整用 GUI を容易に実装できるフレームワークを提供しており、細かなパラメタ調整をしやすくしている。一方で、各オブジェクトの軌跡を表示することはできても、自由に軌跡を編集して動きを指定することは不可能である。これは、オブジェクトの動きを具体的な座標のリストとして保持するか、アルゴリズムとパラメタの組み合わせとして保持するかというアプローチの差であり、両者を融合する手法の開発は将来の課題と考えている。

5.3 パラメタ調整を容易にするプログラミング環境

Adobe Flash や After Effects では、ActionScript などのスクリプト言語でアニメーションをコントロールできる。歌詞をアニメーションさせるために、各文字や単語ごとに動きを定義するスクリプトを割り当てることも可能であり、すべてのオブジェクトに対して個別に動きをつけるような手間はかからない。しかし、動きを定義するスクリプトと GUI を橋渡しする仕組みが存在せず、スクリプトで利用するパラメタを調整するために、スクリプトの記述とアニメーションの確認を何度も繰り返す必要がある。

このようなソースコードと実行結果の溝を解決するために、ソースコード中でパラメタだけ特別扱いし、実行結果を見ながらパラメタをインタラクティブに操作できるようにする手法が提案されている。例えば、変数宣言に付与されたコメントを解析してプログラム実行中に当該変数の値を調整できるスライダーを自動生成する手法[5]や、変数の型を解析して、型に応じた適切なパラメタ調整用 GUI を表示する手法[18]がある。いずれの場合もパラメタ調整用 GUI の実装は開発環境側で用意してあり、本手法のように自由に作成・編集することはできない。

また、物理シミュレーションを用いたアニメーション制作やゲーム開発を支援する研究において、パラメタの決定を支援する手法が提案されている。例えば、パラメタを変化させた場合のオブジェクトのアニメーションの軌跡を重ねて表示し、望ましいものを選んで徐々に絞り込んでいける手法[19]や、ゲームを実行したとき特定の瞬間に物体が取っていた位置に対して制約をかけることでパラメタを逆算する手法[20]が提案されている。これらはいずれも望ましい実行結果を選ぶことでパラメタを絞り込む手法であり、パラメタを GUI で変化させて望ましい実行結果を得る本手法と相補的な役割を果たすものである。

6. 議論

6.1 歌詞をより魅力的に伝える動画表現の大衆化

音楽は、音楽と同期した動画と共に鑑賞すればより楽しみが増す。ボーカル付き音楽は、歌詞カードと一緒に楽しめばより深く制作者の意図を理解できる。つまり、文字のメッセージを動画により印象づける Kinetic Typography という表現手法を歌詞に適用すれば、歌詞をより魅力的に伝える音楽動画が実現できるはずである。筆者らは、本研究を通してそのような動画表現の大衆化を目指している。

例えば本研究により、曲は作れるが動画は作れなかった音楽制作者が、歌詞に込めたメッセージを歌詞アニメーションによって直接表現した動画を制作、配信できるようになる。また、そもそも音楽動画では、アーティストを撮影した動画や、イラスト、キャラクターのアニメーションなどが主役となることが多く、歌詞はともすれば目立たないよう動画のセーフエリア（辺縁部）に追いやられ、カラオケ字幕のように単調に表示されがちであった。そこで本研究が、音楽動画制作者の持つ表現のレパートリーを広げる役割を果たせることを期待している。

時刻同期コメント付き動画配信サービスのニコニコ動画[*7]で見られる「コメントアート」には、歌詞をより魅力的に伝えたいという欲求に応えてきた側面がある。コメントアートとは、動画上に時刻同期コメントを多数投稿し、配信されている動画に対して単なる文字情報を超えた視覚表現を付与する表現手法である。これにより、楽曲の歌詞を凝った色やレイアウトで表示する例がしばしば見られる。ただし、コメントアートは他者がコメントを投稿すると徐々に消えてしまい、文字の複雑な動きをコントロールすることもできない。本研究はコメントアートを置き換えるものではないが、前述のような制約がなく、より自由度の高い表現手段と言える。

6.2 アニメーションの N 次創作

YouTube[*8]やニコニコ動画といった動画配信サービスで

[*7] ニコニコ動画. http://nicovideo.jp/

[*8] YouTube. http://www.youtube.com/

は、鑑賞した動画を自分なりに改変することが容易ではない。一方、TextAlive で作成されたアニメーションは、見た目の複雑さとは裏腹に JSON フォーマットで簡潔に表現され、容易に派生作品を制作できるように設計されている。

例えば、不満点があれば、すぐに自分なりの演出を試してみることができ、思い通りのアニメーションに改変できる。また、替え歌に対して、同じようなスタイルの動画を一瞬で作成できる。派生作品が制作されることを期待する楽曲制作者にとっては、JSON ファイルや歌詞ファイル（通常のテキストデータや発声タイミングの情報が付与されたフォーマットなど）を公開するインセンティブが高まり、さらに TextAlive のような音楽動画制作システムが重用されるサイクルが生まれる可能性がある。

6.3 歌詞以外への応用と他ツール連携

Kinetic Typography を利用した動画は、音楽ではなく朗読を対象にしていることもある。TextAlive は、現状、歌詞のアニメーション制作しか行えないが、朗読に対して音声認識を行い、認識結果を表示・編集できる既存技術[21]を利用すれば、朗読内容の Kinetic Typography 制作に応用できる。また、すでに朗読の書き起こしが済んでいる録音データを、時間的対応付けを保ったまま編集できるインタフェース[22]が提案されている。このインタフェースでは、不要な息継ぎや言葉の繰り返しなどをテキストエディタで削除すると、録音データも同期して編集される。これは本研究と同様に時間的対応付けを持つマルチメディアを編集する手法であり、相補的な役割を持つため、将来的に統合できる。

Kinetic Typography では、文字以外のオブジェクトが画面上に配置されていることもある。例えば、オブジェクトの周りに文字が流れるような効果が考えられる。このような動画制作をサポートするためにはデータ構造とユーザインタフェースの拡張が必要である。現実的には、Adobe After Effects のような外部ツールにデータをエクスポートする仕組みを用意することで、文字情報のアニメーションだけ本ツールで制作し、その他は汎用ツールに任せるような役割分担が必要だろう。

7. おわりに

本稿で提案した TextAlive は、音楽に連動した歌詞のアニメーションを半自動的に生成でき、直感的な GUI で編集可能なシステムである。研究上の貢献は、1) アニメーションの初期パラメタを自動的に生成することで、既存ツールでは必要だったユーザの膨大な手作業を削減した点と 2) パラメタ調整用 GUI に対する実装支援を提供することで、ユーザがアニメーションを直感的かつ精密に編集しやすくした点である。今後は、本制作環境が既存の音楽関連コンテンツ制作・編集・鑑賞のエコシステムの中に馴染むよう拡張を行っていきたい。

謝辞 本研究の一部は JST CREST「OngaCREST プロジェクト」の支援を受けた。

参 考 文 献

[1] Adobe After Effects, http://www.adobe.com/After Effects/.
[2] Norman, D.A.: The Psychology of Everyday Things, Basic Books, Inc., New York (1988).
[3] Fujihara, H., Goto, M., Ogata, J. and Okuno H.: LyricSynchronizer: Automatic Synchronization System Between Musical Audio Signals and Lyrics, IEEE Journal of Selected Topics in Signal Processing, Vol.5, No.6, pp.1252-1261 (2011).
[4] Goto, M.: A Chorus-Section Detection Method for Musical Audio Signals and Its Application to a Music Listening Station, IEEE Transactions on Audio, Speech, and Language Processing, Vol.14, No.5, pp.1783-1794 (2006).
[5] Hartmann, B., Yu, L., Allison, A., Yang, Y. and Klemmer, S.: Design As Exploration: Creating Interface Alternatives through Parallel Authoring and Runtime Tuning, Proc. of. UIST 2008, pp.91-100.
[6] McDirmid, S.: Usable Live Programming, Proc. of SPLASH Onward! 2013, pp. 53-62.
[7] Lee, J., Jun, S., Forlizzi, J. and Hudson, S.: Using Kinetic Typography to Convey Emotion in Text-based Interpersonal Communication, Proc. of DIS 2006, pp.41-49.
[8] Chao, C. and Maeda, J.: Concrete Programming Paradigm for Kinetic Typography, Proc. of IEEE VL 1997, pp.446-447.
[9] Lee, C. J., Forlizzi, J. and Hudson, S.: The Kinetic Typography Engine: Extensible System for Animating Expressive Text, Proc. of UIST 2002, pp.81-90.
[10] Forlizzi, J., Lee, J. and Hudson, S.: The Kinedit System: Affective Messages Using Dynamic Texts, Proc. of CHI 2003, pp.377-384.
[11] Jason, L.: Writing-Designing-Programming: The NextText Project, Media-Space Journal: Special Issue on Futures of New Media Art, Vol.1, No. 5, Media-Space Perth (2008).
[12] Minakuchi, M. and Tanaka, K.: Automatic Kinetic Typography Composer, Proc. of ACE 2005, pp.221-224.
[13] Strapparava, C. and Valitutti, A.: Bringing the Text to Life Automatically, Proc. of AAAI 2006 Workshop on Computational Aesthetics.
[14] aescripts: TypeMonkey. http://aescripts.com/typemonkey/
[15] Davis, R. C., Colwell, B. and Landay, J. A.: K-Sketch: A 'Kinetic' Sketch Pad for Novice Animators, Proc. of CHI 2008, pp.413-422.
[16] Santosa, S., Chevalier, F., Balakrishnan, R. and Singh, K.: Direct Space-Time Trajectory Control for Visual Media Editing, Proc. of CHI 2013, pp.1149-1158.
[17] Kazi, R. H., Chevalier, F., Grossman, T., Zhao, S. and Fitzmaurice, G.; Draco: Bringing Life to Illustrations with Kinetic Textures, Proc. of CHI 2014, pp.351-360.
[18] Kato, J. and Igarashi, T.: VisionSketch: Integrated Support for Example-Centric Programming of Image Processing Applications, Proc. of GI 2014, pp.115-122.
[19] Twigg, C. D. and James, D. L.: Many-Worlds Browsing for Control of Multibody Dynamics, ACM Trans. of SIGGRAPH 2007, Vol.26, No.3 (2007).
[20] Ha, S., McCann, J., Liu, K. and Popovic, J.: Physics Storyboards, Computer Graphics Forum (Proc. of Eurographics 2013), Vol.32, No.2, pp.133-142 (2013).
[21] Goto, M. and Ogata, J.: PodCastle: Recent Advances of a Spoken Document Retrieval Service Improved by Anonymous User Contributions, Proc. of Interspeech 2011, pp.3073-3076.
[22] Rubin, S., Berthouzoz, F., Mysore, G. J., Li W. and Agrawala, M.: Content-Based Tools for Editing Audio Stories, Proc. of UIST 2013, pp.113-122.