TextAlive: インタラクティブでプログラマブルな Kinetic Typography 制作環境

加藤 淳　中野 倫靖　後藤 真孝*

概要．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，文字に対して一から動きを設計するか既定のテンプレートの範囲内で動きを選ぶ必要があった．本環境では，楽曲と歌詞から自動生成した初期アニメーションを，歌詞に特化したインタフェースでインタラクティブに編集できる．また，動きのテンプレートを編集でき，そのパラメタ調整用に制作環境の GUI を拡張できる．

1　はじめに

音楽と文字の芸術は，ライブ演奏や写本による一回性の芸術であったものが，蓄音機や活版印刷により情報としての複製が容易となった後，映像メディアの登場によって映像情報とともに楽しまれるようになった．歌詞は音楽に含まれる文字情報であり，その表現手法は双方の影響を受けてきた．具体的には，古くは口頭伝承され，歌詞カードのような印刷物に記述されていたものが，ビデオカラオケや音楽動画中では発声タイミングに合わせて動画上に表示されるようになった．その後，歌詞を動画上でより魅力的に表現するために，活字を美しく読みやすく配列する技芸である Typography を動画に応用した表現手法 Kinetic Typography が用いられた．例えば，歌詞が音楽に合わせて派手にアニメーションする音楽動画が数多く制作されている．

しかし従来は，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かった．動画を一から制作できる汎用動画制作ツール（Adobe After Effects [1]など）では，各文字の見た目（表示位置や大きさ，字形など）と時間（表示開始時間，終了時間，目立たせるタイミングなど）に関する様々なパラメタを手作業で指定する膨大な手間が必要だった．Kinetic Typography に特化したツールでは，文字の動きを設計する過程がテンプレート選択および少数のパラメタ指定により簡略化されるが，歌詞の発声タイミングとの時間合わせは手作業だった．また，予め用意されたテンプレートを超えた映像表現はできなかった．

このように，歌詞が楽曲と同期してアニメーションする動画は音楽の楽しみ方を広げてくれる一方で，現状，1）制作過程が膨大な手作業で支えられており，2）初心者がテンプレートを超えた思い通りの表現を行うことが難しかった．そこで本稿では，当該動画に特化した制作環境 TextAlive（**図 1**）を提案する．TextAlive は，先述した 2 つの問題を解決するために 1）混合音中の歌声と歌詞の自動対応付け技術を活用して初期アニメーションを自動生成でき，生成結果を歌詞ならではの特徴を利用して編集できる GUI を備えた**インタラクティブな制作環境**である．また，2）テンプレート編集を容易にするため動きのテンプレート開発環境を内蔵し，テンプレートのパラメタ調整用に制作環境の GUI を拡張できる**プログラマブルな制作環境**である．システムの動作する様子や被験者実験の作例は，Web に掲載した[2].

2　TextAlive による Kinetic Typography 制作

本章では，TextAlive による Kinetic Typography の制作フローを紹介する（**図 2**）．まず 2.1 節では，前提知識を有しない動画制作者でも容易に利用できる，動画の自動生成とインタラクティブな編集用ユーザインタフェースについて紹介する．続く 2.2 節では，初歩的なプログラミングの知識を有する制作者を対象とし，文字の動きのテンプレートを Live Programming できるプログラマブルなユーザインタフェースについて紹介する．

2.1　Kinetic Typography 動画の自動生成と編集

TextAlive では，歌声を含む楽曲（混合音）と歌詞を入力すれば，再生可能な動画を自動生成できる．従来は，動画制作用ツール等で歌詞表示のタイミン

図 1. TextAlive の概観

Copyright is held by the author(s).
*Jun Kato, Tomoyasu Nakano, Masataka Goto, 産業技術総合研究所

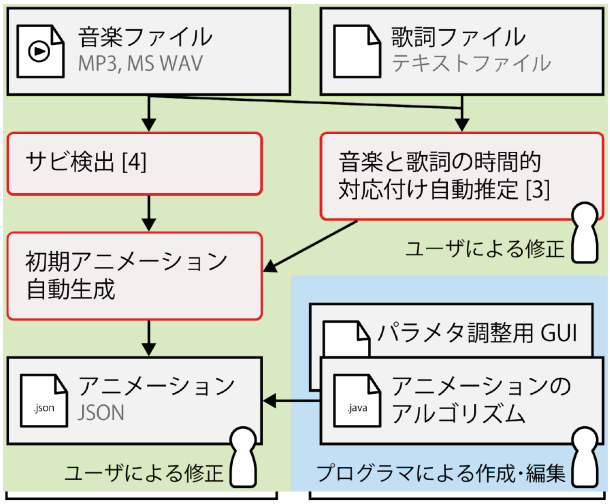

図 2. TextAlive における処理の流れ

グや演出方法などを一から指定する必要があったが，TextAlive では，生成された結果を叩き台として違和感のある部分を修正するだけで，思い通りの演出に近づけることができる．この際，歌詞が**フレーズ**（歌詞中の一行で表される範囲），**単語**，**文字**の階層構造で構成されていることや，サビ部分が同じ長さで繰り返していることなど，歌詞の特徴を利用したインタラクションにより効率的に演出を行える．

2.1.1 動画の自動生成と発声タイミングの修正

ユーザは，はじめに歌声を含む楽曲の MP3 ファイルと歌詞の文字列を入力する．システムは，入力データから歌声と歌詞の時間的対応付けを推定[3]する．これにより，歌詞中のフレーズ，単語，文字の発声開始・終了のタイミングが仮に決定する．さらに，システムは楽曲のサビ区間を検出[4]し，サビ区間が他より豪華になるようなアニメーションを自動生成する．これ以降，ユーザはいつでもタイムライン（**図 3**）の再生ボタンをクリックしたりシークバーを操作したりして，ステージ（**図 1**）でアニメーションの生成結果を確認できる．通常の再生操作では楽曲全体が再生されるが，タイムラインで特定の区間を選択した状態では区間リピートを行える．

発声タイミングの推定結果が誤っている場合，ユーザはタイムラインでフレーズや単語，文字をドラッグ&ドロップしてタイミングを修正できる．タイムライン最上列には楽曲中の歌声のみを抽出した音の波形が表示されており，修正時に参考にできる．また，編集パネルのボタン操作により，選択範囲のタイミングを再推定させたり（**図 4a**），前・後方に詰めたり，等間隔に配分したりできる（**図 4b**）．当該範囲のタイミング情報をコピーしてから，他の同じ構造を持った範囲を選択してペーストすることで，サビが繰り返す場合などにタイミング修正の労力を

図 3. TextAlive のタイムライン

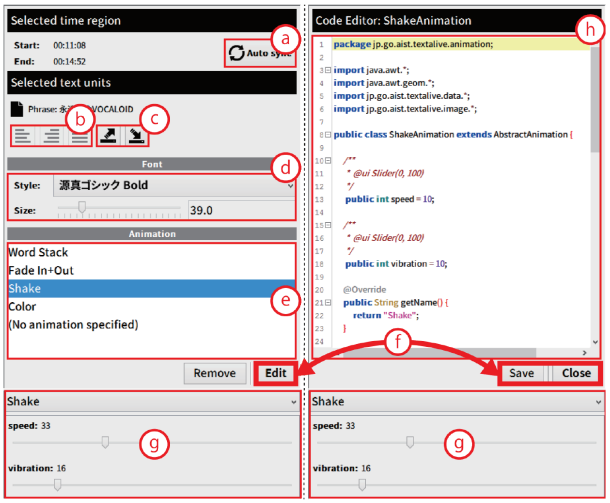

図 4. TextAlive の編集パネル

大幅に削減できる（**図 4c**）．範囲選択は，タイムライン上でドラッグ&ドロップして矩形を描くことで行える他，タイムライン上のサビ情報をクリックすることで，サビ区間だけを選択できる．

なお，発声タイミングを修正すると，アニメーションは再生中であっても即座に更新される．修正した情報は自動的に保存され，次に同じ組み合わせの楽曲と歌詞を選択した際に利用される．

2.1.2 フォントと動きのテンプレートの差し替え

自動的に生成されたアニメーションは決め打ちのアルゴリズムで文字ごとにフォントや動きが割り当てられており，ユーザの演出意図は一切反映されていない．そこで，ユーザは，タイムラインでフォントや動きを変えたい範囲を選択し，編集パネルでフォントの種類や大きさ（**図 4d**）および動きのテンプレート（**図 4e**）を選択する．

ここで，フォントは一文字単位で調整する必要がなく，例えばフレーズを選択してフォントを変えたら，フレーズに含まれるすべての文字についてフォントが変更される点に注意されたい．また，動きのテンプレートは，フレーズ，単語，文字それぞれについて割り当てられる種類が異なり，また，複数割

り当てることが可能である．例えば，フレーズに「画面左から右へスライドするテンプレート」を割り当て，各文字に「発声時に文字が上方へ跳ねるテンプレート」を割り当てられる．追加で，各文字に対して「発声箇所が赤色になるカラオケ表示テンプレート」を割り当てることもできる．このように，複数のテンプレートを自由に組み合わせて複雑なアニメーション効果を設計することが可能となっている．

2.1.3 動きのテンプレートごとのパラメタ調整

各テンプレートは演出内容をカスタマイズできる固有のパラメタを持っており，編集パネルまたはステージ上にパラメタ調整用の GUI ウィジェットが表示される（**図 4g**）．ここで利用できる GUI ウィジェットの種類については**2.2.2項**で後述する．

ユーザは，これらのユーザインタフェースを用いてテンプレートの動きをカスタマイズし，自身の演出の意図を反映させられる．なお，パラメタ調整では，動きがどう変化するかインタラクティブに確認できることが望ましい．しかし，通常，動きの変化は動画を再生しないと確認できず，パラメタを調整するたびに区間リピートするのは非効率である．そこで，ステージ上で選択中のフレーズ・単語・文字が前後 1 秒に移動する軌跡を表示し，ユーザが時間をかけず動きを設計できるようにしている（**図 5**）．

2.2 文字の動きの Live Programming

一般に，動画制作においてプログラミングの知識は必須ではない．しかし，汎用動画制作ツールの多くがスクリプトエンジンを搭載しており，プログラミングによって手作業以上の表現力を引き出せる．例えば，大量のオブジェクトに対し精密な動きを指定できる．しかし，従来のエンジンは，様々なパラメタを持つ動きのアルゴリズムを定義して何度も再利用したり，パラメタをインタラクティブに操作して動きを調整したりすることが容易ではない．

そこで TextAlive は，動きのテンプレートをプログラミングできるだけでなく，その内容に関するグラフィカルなフィードバックを随時得られ，パラメタの調整を GUI で行える Live Programming を実現する．TextAlive を利用するにはプログラミングの知識は必須でないが，もしユーザがその知識を有する場合には，動きのテンプレートのプログラミングによる編集と GUI 操作でのパラメタ調整をボタン操作でスムーズに行き来して両方の利点を享受できる（**図 4f**）．

図 5. 文字の軌跡の表示例（○: 現在位置，×: 1 秒前後の位置）

2.2.1 動きのテンプレートの実装

ユーザは，テンプレートを選択した状態で編集ボタンをクリックし，テンプレートのソースコードを編集するためのコードエディタを表示できる（**図 4h**）．各テンプレートはプログラミング言語 Java の特定のインタフェースを実装したクラスとして定義されており，ユーザはソースコードを書き換えて保存ボタンをクリックすることで，テンプレートの動作を更新できる．テンプレートの保存はいつでも可能であり，例えば楽曲を再生しながらテンプレートを書き換えて結果を確認できる．なお，エディタ中でクラス名を変更して保存ボタンをクリックすると，現在の実装を新しいテンプレートとして保存できる．

動作の更新時はフレーズ，単語，文字ごとに調整したパラメタが初期値に戻ることのないよう，なるべく保持される．通常，プログラムの実装を書き換えて更新内容を確認するには，プログラム停止時に状態情報を保存し，プログラムの再起動時には状態を復元するコードを自力で記述する必要があるが，本環境ではこれらの処理を透過的に実行する．これにより，ユーザはアルゴリズムの実装に集中できる．

2.2.2 パラメタ調整用 GUI ウィジェットの実装

2.1.3項で述べたように，各テンプレートは，演出内容をカスタマイズできるようパラメタ調整用の GUI ウィジェット（**図 6**）をユーザに提供する．このウィジェットは，テンプレートの実装時に Java クラスの public フィールドに対して *@ui* で始まる書式でアノテーションを行うことで生成される．

GUI ウィジェットは，テンプレートの実装が最終的に固まった後はプログラミングの知識を持たないユーザが利用するカスタマイズ用インタフェースとなるが，実装が固まる前段階でもデバッグやテスト用途として有用である．すなわち，テンプレートを

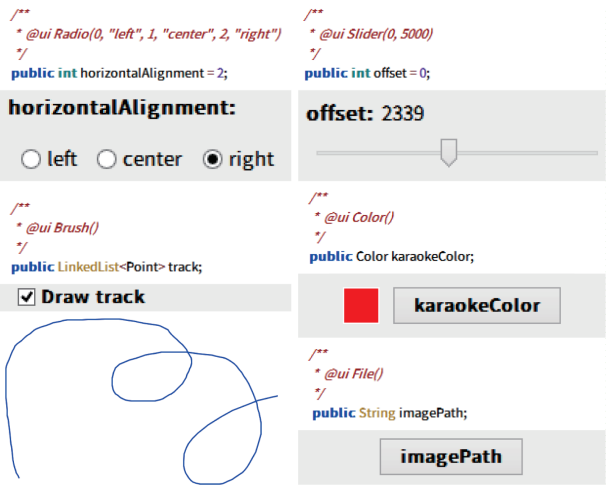

図 6. アノテーションにより自動生成される GUI の例

編集する間，値を変化させてみたいパラメタを仮にウィジェット化して，GUI 操作によって値をいろいろ変化させることで実装の妥当性を確認できる．

3 実装

本章では，前章で紹介した制作フローを実現するための TextAlive の実装手法を提案する．

3.1 概要と動作環境

TextAlive における処理の流れは**図 2**に示した通りである．本システムは Java 7 で実装されており，64bit 版 Windows 8.1 上で動作確認を行っている．入力する楽曲の MP3 ファイルは内部で FFmpeg を用いてモノラルの MS WAV にデコードされる．また，歌詞ファイルは MeCab によって分かち書きされ，平仮名，漢字，アルファベットが全てカタカナの読みに変換される．これらの情報をもとに，歌声と歌詞の時間的対応付け推定[3]とサビ検出[4]が行われ，歌詞アニメーションの Kinetic Typography 動画が自動生成される．Live Programming は，Java Reflection API と Eclipse Compiler for Java を利用して Java クラスを動的にリロードし，オブジェクトの状態情報を保存･復元して実現している．

3.2 動画の自動生成

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，シンプルであるほうがよいと考えられる．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，自動生成の方法もプログラミングできるようにして，ユーザが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り，アニメーションの演出上も，他の区間より盛り上げたほうが曲調と合うと考えられる．そこで，サビ以外についてはシンプルなテンプレート，具体的には各フレーズに「文字が右方向にスライドするテンプレート」と各文字に「発声箇所が赤色になるカラオケ表示テンプレート」が割り当てられる．そして，サビについては各フレーズに「文字が右方向にスライドした後回転しながら拡大するテンプレート」が，各文字にカラオケ表示テンプレートと「発声時に文字が上方に跳ねるテンプレート」が割り当てられる．

3.3 文字のアニメーション

Kinetic Typography 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，複数の単語がフレーズを形成し，フレーズごとに文字の重複を避ける処理が行われたりする．このように，フレーズ，単語，文字の変形や配置を決める処理は，他階層とは独立しており，上位階層を基準にした相対座標系でシンプルに記述できることが多い．

そこで，歌詞のフレーズ，単語，文字を表すオブジェクトは，発声開始・終了タイミングと，親要素の相対座標系での変形内容（移動，回転，拡縮）を表すアフィン変換の行列を保持している．また，色や透明度の情報も保持している．各瞬間での歌詞のレンダリング処理は，まずフレーズ，単語，文字のオブジェクトそれぞれについて，割り当てられたテンプレートに基づき，相対座標系での変形内容や見た目に関するパラメタを計算する．次に，各文字の変形内容や見た目のパラメタを画面上の絶対座標系に変換し，画面内へ収まるものだけが描画される．

各テンプレートを表す Java クラスは，現在時刻を引数として受け取り，自身に割り当てられたオブジェクトの変形内容や見た目のパラメタを操作するメソッドを実装している．

3.4 Live Programming

テンプレートを表す Java クラスは，TextAlive 起動時にソースコードがコンパイルされ，Java VM に動的に読み込まれる．また，アニメーション制作中にユーザがソースコードを編集した際は，クラスが再度コンパイルされ，リロードされる．通常の Java 用開発環境では，プログラム起動中にクラスをリロードする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，新たにテンプレートが割り当てられる際に新しい実装が利用されるだけで，すでにテンプレートを割り当て済みの文字の動きは更新されない．そこで，**3.5節**で詳述する方法により，可能な限り古いテンプレートのパラメタを保ったまま新しいものに差し替え，動きを更新する．こうして，ユーザがシームレスにプログラムの実行（動画編集）とソースコードの編集（テンプレート実装）を行き来できる Live Programming [5]を実現する．

また，ソースコードをコンパイルする際，同時に静的コード解析が行われ，パラメタ調整用 GUI ウィジェットを生成するための情報が収集される．具体的には，Java クラスの public フィールドに付与された *@ui* で始まるコメントが Java 言語のサブセットのインタプリタによって解釈され，**図 6**のような GUI が生成される．インタプリタを用いることで，制作中の動画の幅が上限値となるスライダーなど動的な値を利用した GUI を生成できる．

3.5 動画の保存と読み込み

TextAlive で作成された Kinetic Typography 動画を保存するには，歌詞を構成する文字，単語，フレーズ）を時系列で並べ，それぞれについて表示すべき文字，フォント，音楽との時間的対応付け（発声

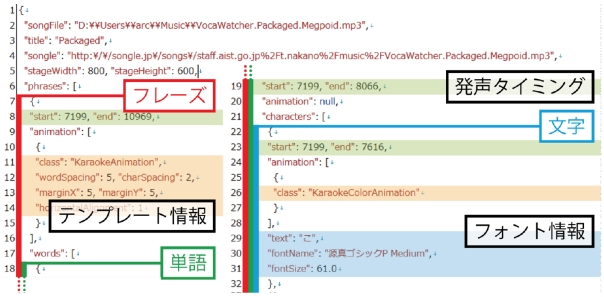

図 7. Kinetic Typography 動画を表す JSON ファイルの例（抜粋）

開始と終了のタイミング）および割り当てられたテンプレートの種類とパラメタを書き出せばよい．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．アニメーションを読み込む際は，この JSON オブジェクトをもとに文字やテンプレートの情報を復元すればよい．

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，テンプレートの実装が更新されている可能性がある．そこで，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．このような場合は，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で，可能な限り元のパラメタを保持するように処理する．

4 関連研究

4.1 Kinetic Typography

Kinetic Typography は，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[6]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，専用のプログラミング言語[7]やフレームワーク[8]が提案されている他，GUI で文字列に動きをつけられるエンドユーザ向けツール[9]が提案されてきた．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生成する手法[10]も提案されている．汎用動画制作ツールにおいて，予め用意された動きのテンプレートから好きなものを選ぶと Kinetic Typography 動画が生成されるプラグイン[11]が発売されている．

上記の 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全般を扱っている．一方，本研究は歌詞の Kinetic Typography 動画を対象とし，歌詞の特徴を活かして労力を削減するインタラクティブなユーザインタフェースを提供する．

4.2 アニメーション制作ツール

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，時間軸上にキーフレームを指定すると間のフレームが滑らかに補間され，ア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，ペンやタッチ入力でオブジェクトの軌跡を描くインタラクション手法[12]が提案されている．また，ペン入力で多くのオブジェクトが似た傾向を持って動くアニメーションを描けるインタフェース[13]が提案されている．しかし，キーフレームのように動きを細かく調整することは難しい．

本研究では，個々の文字でなく単語やフレーズに対して動きのテンプレート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，テンプレートごとにパラメタ調整用 GUI を提供しており，細かな調整がしやすい．

4.3 パラメタ調整を容易にする開発者用ツール

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制御できる．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を特別扱いし，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可能にする手法が提案されている．例えば，変数宣言に特定のコメントが付与されていたら，プログラム実行中に変数値を調整できるスライダーを自動生成する手法[14]や，画像処理コンポーネントへの入力画像上に矩形や円などを描画し，コンポーネントで利用するパラメタを動的に調整できる手法[15]がある．いずれもパラメタ調整用 GUI の種類を指定できず，本手法のように適切なものを選んで動的なパラメタで柔軟に生成することはできない．Unity [16]は C#クラスのフィールド値を調整するための GUI を柔軟に生成できるが，通常のコードを記述する必要があり，コメント一行では済まない．

5 議論および今後の課題

Kinetic Typography は音楽でなく朗読を対象とした動画でも利用されている．朗読に対して音声認識を行い，認識結果を表示・編集できる技術[17]を利用すれば，TextAlive を楽曲のみならず朗読にも応用できる．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[18]が提案されている．このインタフェースでは，テキストエディタ上で書き起こしを削除すると録音データ上の対応箇所が削

除される．TextAlive でも同様に，動画を編集すると楽曲が適切に編集される機能を実現できるだろう．

我々は，動画制作またはプログラミングを仕事や趣味で行った経験のある 7 名の被験者に, 2-3 時間, 朗読にも対応させた TextAlive を試用してもらった [2]．全員が好みの楽曲または朗読について 30 秒程度の Kinetic Typography 動画を制作でき，タイミング自動推定機能が好評を博し実用性を確認できた一方，プログラミング用インタフェースの改善およびテンプレート拡充が必要なことが分かった．

音楽や動画配信サービスの興隆と共に，音楽の制作手段が大衆化し，多くの人が自己表現できるようになっている．しかし，音楽に同期した動画を制作することは未だ容易ではない．動画配信サービスで公開される音楽動画では，アーティストを撮影した動画やイラストなどが主役となり，歌詞は動画のセーフエリア（辺縁部）に目立たないよう表示されることが多く，表示すらされないこともある．Kinetic Typography は歌詞が持つメッセージを視覚的に印象付けることができる表現手法であり，楽曲と歌詞のみから魅力的な音楽動画を制作できる. 本研究は, Kinetic Typography を大衆化することで, 音楽制作者の表現の幅を広げられると期待している．なお, 現在の実装では，アフィン変換や透明度の操作などを用いて Kinetic Typography のテンプレートを開発できる．今後カメラ座標系を導入し，文字が三次元空間で動き回るような演出にも対応予定である．

我々は，TextAlive を Web 上で公開予定である．現状すでに多数の楽曲と歌詞が Web 上で公開されているため，これらを用いてユーザがすぐに動画制作を始められるようになる．さらに，クラウドソーシングで楽曲の正しい発声タイミングを蓄積できる．一般に，動画配信サービスで配信されている動画は視聴者が改変しづらいが，TextAlive では動画が編集容易な JSON で保存されるため, 派生動画や替え歌動画などの N 次創作が容易に制作可能となる．

謝辞 本研究の一部は JST CREST の支援を受けた．

参考文献

[1] After Effects. http://www.adobe.com/After Effects/

[2] TextAlive. https://staff.aist.go.jp/jun.kato/TextAlive/

[3] H. Fujihara, et al.: LyricSynchronizer: (略), IEEE Journal of STSP, Vol.5, No.6, pp.1252-1261 (2011).

[4] M. Goto: A Chorus-Section Detection Method for Musical Audio Signals and Its Application to a Music Listening Station, IEEE Trans. on ASLP, Vol.14, No.5, pp.1783-1794 (2006).

[5] S. McDirmid: Usable Live Programming, Proc. of SPLASH Onward! 2013, pp. 53-62.

[6] J. Lee, et al.: Using Kinetic Typography to Convey Emotion in Text-based Interpersonal Communication, Proc. of DIS 2006, pp.41-49.

[7] C. Chao, and J. Maeda: Concrete Programming Paradigm for Kinetic Typography, Proc. of IEEE VL 1997, pp.446-447.

[8] J. Lee, et al.: The Kinetic Typography Engine: (略), Proc. of UIST 2002, pp.81-90.

[9] J. Forlizzi, et al.: The Kinedit System: (略), Proc. of CHI 2003, pp.377-384.

[10] M. Minakuchi, and K. Tanaka: Automatic Kinetic Typography Composer, Proc. of ACE 2005, pp.221-224.

[11] TypeMonkey. http://aescripts.com/typemonkey/

[12] R. Davis, et al.: K-Sketch: (略), Proc. of CHI 2008, pp.413-422.

[13] H. Kazi, et al.; Draco: (略), Proc. of CHI 2014, pp.351-360.

[14] B. Hartmann, et al.: Design As Exploration: (略), Proc. of. UIST 2008, pp.91-100.

[15] J. Kato, and T. Igarashi: VisionSketch: (略), Proc. of GI 2014, pp.115-122.

[16] Unity. http://unity3d.com/

[17] M. Goto, and J. Ogata: PodCastle: (略), Proc. of Interspeech 2011, pp.3073-3076.

[18] S. Rubin, et al.: Content-Based Tools for Editing Audio Stories, Proc. of UIST 2013, pp.113-122.

未来ビジョン

統合開発環境（IDE）はプログラム開発の機能が揃った便利な環境である. しかし, Vim や Atom などのテキストエディタにスクリプトを組み込んで自分が真に使いたい機能を実装する人が数多くいる．エディタのほうがいじりやすく手に馴染むのだろう．このいじりやすさを「**プログラマビリティ**」と呼ぶ．

では，創作支援ツールはどうか．様々なツールにスクリプト言語が用意され，手作業を超えた制作効率を実現できるが，本質的な機能向上には別の IDE でプラグイン開発が必要なことが多く, **プログラマビリティ**が低い．

デジタルツールを駆使する次世代クリエイターに必要なのは, **ユーザが機能拡張を行い, 知的生産の効率を向上できる自己進化可能な道具**だと考えている．そのために，ツール設計者は**プログラマビリティ**を適切に（本質的な機能は壊せないよう，しかし柔軟に拡張できるよう）設計しておく必要がある．本研究は, Kinetic Typography 制作ツールについてこの未来ビジョンを目指した一例である．