情報処理学会論文誌 Vol.66 No.2 273–285 (Feb. 2025)

推薦論文

インタラクティブな歌詞駆動型視覚表現「リリックアプリ」開発用フレームワークの提案と実証研究

加藤 淳[1,a)] 後藤 真孝[1,b)]

受付日 2024年5月17日, 採録日 2024年8月1日

概要：音楽に合わせてタイミングよく歌詞が動くリリックビデオは楽曲のプロモーション手段として一般化したが，視聴者は再生操作でしかインタラクションできない．そこで我々は，ユーザとのインタラクションにより歌詞のテキストを再生のたびに異なる方法で提示でき，静的メディアの制約を取り払える歌詞駆動型のインタラクティブな視覚表現を「リリックアプリ」と定義し，この新たな表現形式を開発・配信できる Web ベースのフレームワーク「Lyric App Framework」を提案する．我々は，2020 年に当フレームワークを一般公開して以来，創作文化に関するイベント「マジカルミライ」で毎年開催されるプログラミング・コンテストを通して，新たな表現形式の可能性を探ってきた．本論文では，4 回開催して集まった 159 作品の分析結果と，音楽とプログラミングの未来に関する示唆をあわせて報告する．

キーワード：ツールキット，創造性支援，マルチメディア，音楽同期，“In the wild” 研究

A Framework for Developing Interactive Lyric-driven Graphical Applications “Lyric Apps” and Its Empirical Research

Jun Kato[1,a)] Masataka Goto[1,b)]

Received: May 17, 2024, Accepted: August 1, 2024

Abstract: Lyric apps, as we name them, are a new form of lyric-driven visual art that renders different lyrical content depending on user interaction and address the limitations of static media. To open up this novel design space for programmers and musicians, we present “Lyric App Framework,” a framework for building interactive graphical applications that play musical pieces and show lyrics synchronized with playback. We have held four editions of the annual programming contest since 2020 and collected 159 examples of lyric apps, enabling us to reveal eight representative categories, improve the API design, and report findings.

Keywords: toolkit, creativity support, multimedia, music synchronization, “in the wild” research

1. はじめに

歌詞はメディア技術の進歩とともに多様な方法で聴衆に届けられてきた．たとえば，レコードやコンパクトディスク（CD）には，グラフィックデザイナがデザインした歌詞カードがついてきた．また，音楽動画や動画共有サービスが一般化してからは，楽曲再生に合わせて歌詞が魅力的に動く「リリックビデオ（lyric video）」が人気を博すようになった．リリックビデオは，多くの場合，モーショングラフィックデザイナが歌詞の動きを手付けしているが，我々は，音楽理解技術を活用してその制作を省力化し，支援するサービス「TextAlive」[1] を研究，開発，運営してきた．そして今や，ミュージシャンは多様なメディアを横断して聴衆に楽曲を届けるようになり，聴衆は楽曲を聴くだけでなく，ソーシャルネットワーキングサービス（SNS）への投稿などを通して自らの体験を共有するようになった（参

[1] 産業技術総合研究所
National Institute of Advanced Industrial Science and Technology (AIST), Tsukuba, Ibaraki 305–8568, Japan
[a)] jun.kato@aist.go.jp
[b)] m.goto@aist.go.jp

本論文の内容は 2024 年 3 月のインタラクション 2024 で報告され，同プログラム委員長により情報処理学会論文誌ジャーナルへの掲載が推薦された論文である．

図 1 「リリックアプリ」は，ユーザが歌詞駆動の視覚表現とインタラクションできるようにすることで，リリックビデオの限界を突破する．我々は，プログラマとミュージシャンによるリリックアプリ開発を支援する「Lyric App Framework」を提案する

Fig. 1 "Lyric apps" enable users to interact with lyric-driven visual art synchronized with music, addressing the limitations of lyric videos. We propose "Lyric App Framework" that helps programmers and musicians develop lyric apps.

加型文化 [2]）．また，スマートフォンの普及により，多くの人々が楽曲との斬新なインタラクションを体験できるようになってきた [3]．

こうした社会・文化・技術的背景をふまえ，我々は本論文において，歌詞カードやリリックビデオに続く歌詞駆動メディアとして，楽曲と歌詞を目と耳でインタラクティブに楽しめる「リリックアプリ（lyric app）」（**図 1**）を定義する．さらに，この新たな表現形式の可能性を探るため，プロダクションレディ（production-ready）なリリックアプリを構築するための一貫した開発体験を提供する Web ベースのフレームワーク「Lyric App Framework」を提案する．なお，ここでプロダクションレディとは，プロトタイプ開発だけでなく一般ユーザによる利用にも耐える高い信頼性でのアプリの配信までサポートすることを指す．

本研究の貢献は以下の 3 点である．

(1) リリックビデオを拡張して，楽曲と歌詞に関するインタラクションを可能にする仕組みを加え，新たな表現形式「リリックアプリ」を定義したこと．

(2) Web ベースのワークフローと API（TextAlive App API [*1]）を提供する新たなフレームワークを実装し，プロダクションレディなリリックアプリを構築するための一貫した開発体験を提供したこと．

(3) フレームワークを一般公開[*2]し，創作文化に関するイベント「マジカルミライ」でプログラミング・コンテストを 4 年間開催し，応募 159 作品をもとにリリックアプリの 8 種類のカテゴリを明らかにし，フレームワークの有効性を評価し，今後の研究への示唆を得たこと．

なお，本論文は，国際会議 ACM CHI 2023 での発表論文 [4] を和訳して要約したうえで，2022–2023 年のコンテスト 107 応募作品に関する追加分析と，提案にとどまっていた時区間駆動型 API を実際に実装した報告と考察を加えて発展させたものである．

[*1] https://github.com/TextAliveJp/textalive-app-api
[*2] https://developer.textalive.jp

2. リリックアプリ

リリックアプリの特性を浮き彫りにするために，2.1 節でユーザ「アリス」が典型的なリリックアプリを体験する様子を描き，プログラマが直面する課題を具体的に理解する助けとする．それに基づいて，2.2 節でそうした課題を議論する．

2.1 典型的なリリックアプリの例

ある日の午後，アリスはお気に入りのミュージシャンの新しい SNS 投稿を見つける．投稿内のリンクをたどると Web ブラウザがリリックアプリを起動し，新しくリリースされた楽曲が流れ始める．楽曲のジャケット画像とタイトルが 3D CG 空間内に浮かび上がる．

静的なリリックビデオとは異なり，アリスは画面をタッチして空間内を自由に動き回り，楽曲のテーマに合ったさまざまな視覚的オブジェクトを見つけることができる．そうしたオブジェクトはビートと同期してアニメーションしており，目を楽しませてくれる．歌詞が発声するタイミングでは，歌詞のフレーズが視界に飛び込んでくる．そして，それを読むだけでなく，タッチ操作でつかんで好きなところに置いておくことができ，1 度きりの情景を作りあげることができる．サビではアニメーションの効果が派手になり，気持ちを盛り上げてくれる．

夕方，アリスがふたたびリリックアプリを開くと，同じ 3D CG 空間が夕闇のような暗い色で表示される．歌詞のフレーズも昼間とは違った配置になり，楽曲と歌詞を新鮮に楽しむことができ，楽曲をもっと好きになったように感じられる．さらに，リリックアプリの中で，そのミュージシャンの別の曲を開くこともできる．アリスは，こうしたインタラクションを通して楽曲の世界観により深く浸り，個人的かつ唯一無二な体験を得られる．

2.2 リリックアプリ開発の課題と支援の必要性

上記の筋書きをふまえ，リリックビデオ制作とアプリ開発を比較することで，本節ではリリックアプリを開発する際にプログラマが直面する 3 つの課題と，それぞれに望まれる開発ツールによる支援について考察する．

まず，リリックビデオとリリックアプリを含む歌詞駆動型の視覚表現は，楽曲再生と同期している必要がある．そうした同期を実現するためには，通例，歌詞を含む音楽的要素のタイミング情報を手作業でラベル付け（アノテーション）しなければならない．我々はリリックビデオの制作支援に関する先行研究 [1], [5] において，音楽理解技術による自動解析と，解析結果を修正するためのユーザインタフェースを提案した．こうした半自動的な処理はリリックアプリ開発においても有用だと思われるが，アプリの開発ワークフローの中へどう組み込むかは自明ではない．

次に，ユーザと楽曲および歌詞との効果的なインタラクションデザインは容易ではない．リリックビデオでは対象楽曲が一意に決まっており，時間軸に沿った場面展開も最終的には一案に収束するため，モーショングラフィックデザイナはその完成形に向けて試行錯誤するだけでよい．一方，リリックアプリ開発では，プログラマは，何度実行されても飽きないような視覚効果を生成するアルゴリズムを思いつく必要がある．クリエイティブコーディングのためのライブラリ（3.3 節）の助けは有用だが，リリックアプリ特有の課題として，さまざまな音楽的要素のタイミングや，歌詞の意味，そしてユーザ入力を考慮に入れる難しさは残る．したがって，クリエイティブコーディングの実践を邪魔しないようなツール支援がきわめて重要である．

最後に，リリックアプリをユーザに届ける手段が明らかでない．創造性支援ツールの研究はラボスタディが中心で実用性の面で批判されることも多い [6]．しかし我々は，リリックアプリという新たな表現手段を提案するうえで，ラボ内にとどまらず，プロダクションレディなアプリを開発できるような工学的視点を持つことが重要であると考えている．ミュージシャンが動画共有サービスを使ってリリックビデオを多数の視聴者へ配信できることと同様に，プログラマやミュージシャンがリリックアプリをユーザに届けられるようにすることまで考慮するべきである．

3. 関連研究

本章では，リリックアプリに関連する先行事例や，自動同期技術，マルチメディアのためのプログラミング支援技術を紹介して，本研究の貢献を明確にする．

3.1 リリックアプリに関連する先行事例

視覚と聴覚のメディアが同期することで，没入感のある体験が実現できる．カラオケは，歌詞のシンプルな字幕表示を見ながら，歌詞を正確に覚えていなくても歌を歌って，楽曲を楽しめる娯楽である．機器と楽曲によっては，リリックビデオを再生することで没入感を高める工夫がなされていることもある．リズムゲーム[*3]や，タイピングや外国語を練習する音楽ゲーム[*4]では，楽曲再生と同期して歌詞を表示できるものもある．

これらは歌詞やビートなどの音楽的要素のタイミング情報を利用しており，リリックアプリに関連する先行事例ととらえることができる．しかし，これらは，インタラクションデザインの観点では比較的狭い応用範囲にとどまっている．カラオケはインタラクティビティに乏しく，ゲームはゲームプレイ自体の楽しさやゲーミフィケーションによる学習効果の向上に主眼が置かれている．我々は，リリックアプリには先行事例を超えたポテンシャルがあると考えており，デザイン空間を 5 章で示すように探索するため，リリックアプリ開発を支援するフレームワークを構築した．

3.2 楽曲と歌詞の自動同期技術

視覚表現と楽曲が同期したリリックアプリならではの体験を実現するアプリケーション開発を支援するためには，歌声を含む楽曲（混合音）と歌詞などの音楽的要素を自動対応付けする技術が必要となる．LyricSynchronizer [7] は，楽曲と歌詞の自動対応付けを行い，歌詞テキストをクリックすることで楽曲の再生位置を操作できるユーザインタフェースを提供した．TextAlive [1] は，さらにリリックビデオを制作でき，アニメーション用アルゴリズムをライブプログラミングできる統合制作環境を実現した．SyncPower [8]，Musixmatch [9]，そして LyricFind [10] は，歌詞テキストを取得するための API と，取得したテキストをカラオケのように表示するためのプチリリ機能，FloatingLyrics，そして Lyric Display と呼ばれる視覚的コンポーネントを提供している．

本研究は，ベース手法としている TextAlive を除き，これらの先行事例と 2 つの点で異なる．まず，先行事例では既存の楽曲の情報を取得することしかできないが，本フレームワークでは，ユーザは新たな楽曲と対応する歌詞テキストを自由に登録できる．次に，既存の商用 API では，歌詞テキストを取得できても，タイミング情報に直接アクセスできない．一方，本研究が提供する TextAlive App API では，歌詞のテキストや，歌詞を含むさまざまな音楽的要素のタイミング情報へ直接かつ統一的な方法でアクセスできる．この 2 点の新規性は，プログラマが好きな楽曲のリリックアプリを開発できるようにする鍵となる．

3.3 クリエイティブコーディング

マルチメディアコンテンツを生成したりインタラクション可能にしたりするためのプログラミングを支援するツー

[*3] http://miku.sega.jp/flick
[*4] https://lyricstraining.com

ルは数多く存在する．そうしたツールを用い，芸術的な目的で創造的にプログラミングすることは，しばしば「クリエイティブコーディング（creative coding）」と呼ばれる．

クリエイティブコーディング用のライブラリは Processing [11] のようにネイティブアプリケーションを構築できるものだけでなく，p5.js [*5] のように Web アプリケーションを構築できるものもある．こうしたライブラリは，Web ブラウザを備えた端末で実行できるアプリケーションを容易に開発でき，人気を博している．リリックアプリの視覚表現を実装する助けになるが，プログラマによって好みが分かれる．そこで本研究では，プログラマが好みのクリエイティブコーディング用ライブラリを選べるように，フレームワークを設計している．

なお，我々が研究開発して公開中の Web ベースの大規模音楽連動制御プラットフォーム「Songle Sync」[12] [*6] では，楽曲再生に合わせて多種多様な機器でのマルチメディア演出を同時制御できる．我々は，Songle Sync およびその先行研究である Songle Widget [13] [*7] において，ビートなどの音楽的要素と連動する演出を手軽にプログラミングできるイベント駆動型の API を提供した．ただし，歌詞関連の情報や声量などは利用できない．また，後述（4.2.2 項）のとおり，イベント駆動型の API には高度な演出を開発しづらく，時間精度が落ちやすい問題がある．本研究では，これらの技術的制約を解消する API を新たに提案する．

4. Lyric App Framework

2.2 節で整理したとおり，プログラマはリリックアプリ開発において 3 つの課題に直面する．本章では，各課題を解決する 3 つの主要機能を備えたフレームワーク「Lyric App Framework」を提案する（図 1）．

4.1 Web ベースの開発ワークフロー

まず，楽曲中の音楽的要素と同期した演出開発を容易にするため，Web ベースのワークフローを提案する．このワークフローでは，開発中いつでも新たな楽曲を扱えるようにしたり楽曲を差し替えたりでき，プログラマやミュージシャンのニーズに柔軟に対応できる特長がある．

4.1.1 自動解析とクラウドソーシング

本フレームワークは，我々が研究開発してきた能動的音楽鑑賞サービス「Songle」[14], [15] [*8] の機能を拡張することで，リリックアプリ開発に役立つ情報（**表 1**）を提供し，必要に応じて誤りを修正できるようにする．

プログラマやミュージシャンのワークフローは以下のとおりである．まず，フレームワークの Web インタフェースを通してリリックアプリで利用したい楽曲と歌詞テキストの組を登録する．楽曲と歌詞テキストは新規にアップロードすることもできるし，すでに Web 上にあって二次利用に関する適切なライセンスが付与されているものなら URL を指定するだけでもよい．登録から約 10 分後には，自動解析の結果がフレームワークの Web サーバ上で利用可能になり，リリックアプリ上で読み込めるようになる．

その後，歌詞を含む音楽的要素の自動解析されたタイミング情報を Web インタフェースで確認し，必要に応じて修正できる．こうしたクラウドソーシングによる自動解析結果の修正はすでに Songle 上で可能であったが，我々は本研究や TextAlive の研究で，歌詞タイミングの修正も可能にしたほか，歌詞単語の品詞の自動検出や，タイミング情報のリビジョン管理まで可能な拡張を施した．

表 1 リリックアプリがアクセスできる解析結果の一覧
Table 1 Types of analysis results available to lyric apps.

解析結果	説明
歌詞タイミング	歌詞テキストの各フレーズ，単語，文字が発声を開始し，終了するタイミング情報
品詞	歌詞テキストの各単語の品詞
声量	歌声の音量の時系列変化
音楽印象	音楽の印象の移り変わりを表す V/A（valence arousal）平面上の座標値の時系列変化
ビート構造	四分音符に相当する各ビートや，各小節の先頭のタイミング情報
コード進行	コード名と各コード区間のタイミング情報
楽曲構造	サビ区間や，楽曲中の繰返し区間のタイミング情報

4.1.2 キーとしての楽曲 URL

本フレームワークでは，楽曲の URL が歌詞テキストと解析結果を取得するためのキーとなる．プログラマは，楽曲 URL のリストを保持することでユーザに楽曲を選ばせることができ，単一のリリックアプリで複数の楽曲を開くことが可能となる．ミュージシャンは，未リリースの楽曲について，一般には非公開の暫定 URL でアプリの振舞いを確認し，リリース時には公開 URL に差し替えて，楽曲のプロモーションに役立てることができる．

なお，楽曲 URL だけを指定した場合，解析結果（あるいはユーザによる修正結果）の最新版が読み込まれるが，さらに数値型のリビジョン番号を指定することで，特定の版のデータを取得できる．これにより，クラウドソーシングで仮に悪意ある修正が行われた場合にも影響を受けずに済む．本節のような工夫は，本フレームワークに限らず，静的メディアにインタラクションを追加する今後の研究でも有用であろう．

[*5] https://p5js.org
[*6] https://api.songle.jp/sync
[*7] https://widget.songle.jp
[*8] https://songle.jp

4.2 リリックアプリのインタラクションデザイン用 API

次に，ユーザと楽曲・歌詞の間のインタラクションデザインを支援するため，プログラマ向けの TextAlive App API[*1] を提案する．本 API は楽曲再生の状態を管理できるイベント駆動型の API と，楽曲再生と精密に同期した演出を表示できる時刻駆動型の API に大別される．

4.2.1 状態管理用のイベント駆動型 API

リリックビデオはつねに単一の課題曲と紐づいているが，リリックアプリは単一の楽曲向けのこともあれば，特定の楽曲セットで実行できたり（例：ユーザがプレイする曲目を選べるリズムゲーム），任意の楽曲で実行できたり（例：インタラクティブなビジュアライザのついた音楽プレイヤー）する．さらに，リリックアプリは，特定の操作をユーザに許可したり禁止したりできる．たとえば，音楽プレイヤーは楽曲再生を一時停止したり，任意の再生位置にシークしたりできるが，ゲームはどちらの操作も禁止していることが多い．楽曲の物語性を強調するアプリケーションでは，一時停止は可能でも，シークは禁止されていたりする．

これらすべてのインタラクションシナリオの実装を支援するために，我々はリリックアプリがとりうる状態の一覧を**図 2** のように整理して定義した．本 API では，JavaScript クラス `Player` が，フレームワークの機能を利用するエントリーポイントとなっており，そのインスタンスにイベントリスナを登録すれば，状態間の遷移が通知される．

4.2.2 時間精度を要するインタラクションデザイン用の時刻駆動型（time-driven）API

グラフィカルユーザインタフェース（Graphical User Interface; GUI）用のフレームワークや，我々が研究開発した音楽連動制御のためのプラットフォーム「Songle Sync」[12] では，イベント駆動型の API によって，時系列に発生するイベントに応じたインタラクション設計が行えるようになっている．しかし我々は，リリックアプリ用の API を設計するにあたり，次に述べる 3 つの理由から，**図 3** ⓐに示す音楽的要素のイベントをサポートしないことに決めた．

まず，イベント駆動型の API において，イベントは通常何かが起きた直後にのみ発行されるため，未来のイベントに向けた準備をコーディングする目的には向いていない．たとえば，歌詞の単語の発声開始イベントを使って，発声よりも前に画面内に滑り込み始めるような予備動作をともなうアニメーションの設計には使えない．次に，イベントは個々の音楽的要素に紐づいて提供されるが，魅力的な視覚表現はしばしば複数の音楽的要素と連動している．たとえば，歌詞のフレーズ発声時に周囲に現れる水紋のような視覚表現を考える．水紋はビートに合わせてフェードインとアウトを繰り返すかもしれないし，サビの歌詞かどうかによって色が変わるかもしれない．イベント駆動型 API では，フレーズの発声開始，ビート，サビ区間といったすべてのイベントについてリスナを定義し，情報をどこか別のロジックで集約してアニメーションさせる必要が生じて不便である．最後に，イベント駆動型 API では時間精度が低くなりがちである．こうした API を提供するためには，フレームワーク内にイベントループを設け，現在の再生位置と各種イベントの時刻情報を比較し続けるという実装が典型的である．この場合，各種イベントは最悪値でイベントループの実行間隔分，遅れて発行される可能性がある．このように時間精度を保証できない API は，求める表現を得るうえで不十分なことがあり，時間精度が意図せずに低くなった場合に初学者が混乱する可能性もある．

これらの課題を解決するために，我々は，引数にミリ秒単位の再生位置の数値をとり，その引数で指定された時点での音楽的要素を開始・終了などの時刻情報付きで返す「時刻駆動型（time-driven）API」（図 3 ⓑ）を提案する．たとえば，5 秒後に発声が始まる歌詞フレーズの情報を取得するには `player.getPhrase(player.position + 5000)` という API を呼び出せばよい．さらに，同じ引数で `getBeat`

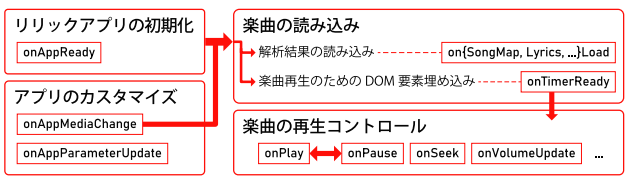

図 2 複雑な状態管理を支援するイベント駆動型 API

Fig. 2 Event-driven API to help manage complex states.

図 3 精密な時刻同期が必要なインタラクションデザインを支援する API の 3 類型

Fig. 3 Three different API designs for helping achieve time-sensitive interactions.

という API を呼ぶなどすれば，フレーズ，ビート，サビ区間の情報をすべて考慮に入れた視覚表現を実装できる．

時刻駆動型 API は，インタラクティブな視覚表現を生成する際に役に立つ，GUI やクリエイティブコーディング用のライブラリと相補的な役割を果たす．こうしたライブラリでは，通常，描画すべき内容を定義するレンダリング用の関数をプログラマが定義するようになっている．たとえば，Web API における `requestAnimationFrame()` [*9]，p5.js [*5]における `draw()`，Three.js [*10]における `renderer.render()` などである．プログラマは，各関数内で時刻駆動型 API を呼ぶことで，複合的な音楽的要素を考慮した視覚表現を設計できる．描画内容は，つねに楽曲再生の最新の状態を反映していることが保証される．このように，本 API はプログラマが好みのライブラリと併用できる特長がある．図 3ⓒで示した API 設計の拡張に関しては 6.1.2 項で後述する．

4.3 リリックアプリの大規模配信

最後に，ユーザにリリックアプリを届ける方法が自明でない課題に関して，本フレームワークは，プログラマやミュージシャンがリリックアプリを Web サイトとして安定的に大規模配信できる仕組みを提供する（**図 4**）．

4.3.1 プロダクションレディなアプリ配信

一般に，インタラクティブなメディアコンテンツの配信は 2 つの観点で難しい．まず，ユーザを取り巻く情報環境は千差万別であり，多くのユーザがインタラクションを楽しめるような相互運用性を担保する必要がある．そして，コンテンツの利用者数は時と場合によって大きく上下するため，一度に大量のアクセスがあった場合などにも対応できるスケーラビリティを確保しなければならない．

そこで我々は，インタラクティブミュージックの大量流通に関する議論 [3] を参考にして，現状最も現実的な解は，

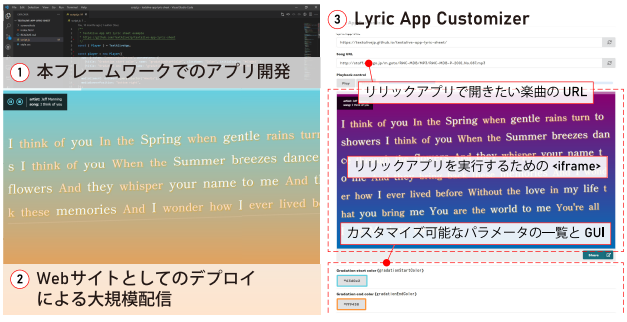

図 4 Web サイトとして大規模配信した後の，事後的なカスタマイズを可能にする Lyric App Customizer

Fig. 4 Lyric App Customizer enables customization of lyric apps after deployment for mass distribution.

[*9] https://developer.mozilla.org/docs/Web/API/Window/requestAnimationFrame

[*10] https://threejs.org

アプリを Web 標準にのっとった Web サイトとして配信できるようにすることだと考えた．本フレームワークを使えば，プログラマが Web サイトを構築する技術さえ持っていれば，上記のような大量流通が可能なプロダクションレディのリリックアプリを実装できる．

4.3.2 Lyric App Customizer

本フレームワークは，プログラマやミュージシャンがリリックアプリの挙動をカスタマイズできるよう，TextAlive Lyric App Customizer [*11]という Web インタフェースを備えている（図 4）．たとえば，4.2.1 項で紹介した音楽プレイヤーのように，楽曲を自由に指定できるリリックアプリの場合，ミュージシャン側で楽曲の URL をカスタマイズできる．カスタマイズ結果は，最終的にはリリックアプリの URL に与えるクエリパラメータとして，`?ta_song_url=NewUrl` のように永続化できる．

また，楽曲 URL の他にも，プログラマは自由にカスタマイズ用パラメータのリストを定義でき，定義されたパラメータの一覧は，Customizer 上でスライダーやカラーパレットなどの GUI として表示される．GUI が操作されてパラメータ値が調整された場合，リリックアプリには `onParameterUpdate` イベントが通知されるため，対応する処理をプログラマが実装すれば，アプリの振舞いを誰でもインタラクティブにカスタマイズできるようになる．ミュージシャンは，Customizer でリリックアプリを自身の楽曲向けにカスタマイズして，プロモーションにつなげることができる．

5. プログラミング・コンテストでの実証実験

我々は 2020 年 9 月 18 日に TextAlive App API を含む Lyric App Framework を一般公開し，年次のプログラミング・コンテストを開催してきた．これは実証実験として，以下の 3 つの目的を持つ．第 1 に，創作文化を楽しみ，創作文化に貢献する人々が集まるイベントでのコンテスト開催により，未来志向のプログラマとともにリリックアプリのデザイン空間を探索する（事例を収集，分析し，リリックアプリの特徴について理解を深める）．第 2 に，本フレームワークが，4 章で述べた設計により，初学者から熟達者まで幅広いプログラマを支援できることを確かめ，よりよい API 設計などの示唆を得る．第 3 に，リリックアプリが，音楽を題材にプログラミングに取り組める新たな機会を生み，プログラマの創作コミュニティ参入を後押しし，同コミュニティを活性化できる，という仮説を検証する．

以降，本章では，プログラミング・コンテストについて紹介するとともに，2020–2021 年の応募作品 52 件を分析して判明したリリックアプリのカテゴリ 8 種類などの結果を報告する．そのうえで，2022–2023 年の応募作品 107 件に関する追加分析と，質的ユーザ評価の結果を紹介する．

[*11] https://developer.textalive.jp/app/customize

5.1 プログラミング・コンテストの概要

プログラミング・コンテストは，年次の大規模イベント『初音ミク「マジカルミライ」』の一環として開催されてきた．本イベントは著名な歌声合成ソフトウェアでありバーチャル・シンガーである「初音ミク」を生み出したクリプトン・フューチャー・メディア株式会社が主催し，「初音ミクたちバーチャル・シンガーの 3DCG ライブと，創作の楽しさを体感できる企画展を併催したイベント」[*12]である．

5.1.1 コンテストのルールと進め方

コンテストは「マジカルミライ」イベント会場での入選作品および我々による注目作品の紹介と受賞作品の発表を除けばオンラインで進行した．主催側でリリックアプリの開発に必要な権利処理が済んでいる楽曲と歌詞を課題曲として提供することで，参加者が開発に集中できるようにした．コンテストの応募期間中，5.1.2 項で後述する参考資料をオンラインで提供し，X（旧 Twitter）[*13]や Gitter [*14]で参加者からの質問に適宜回答した．コンテストの応募に際して（参加は無料），参加者は静的 Web アプリケーションのソースコードとビルド手順を提出した．サーバ側のプログラムを含まない静的アプリケーションに限ったのは，応募作品を主催側の Web サーバに設置して誰でも楽しめるようにするためである．

各年，応募を締め切ってからは，主催側の 3 名または 4 名の体制（コンテストを主催するクリプトン・フューチャー・メディア株式会社の代表取締役である伊藤博之氏と，産業技術総合研究所の我々を含む）で審査し，入選作品を選出した．審査時の評価軸は，クリエイティビティ（Web アプリケーションの演出が楽曲と同期して魅力的に見えるか），イノベーション（Web アプリケーションを支えるアイデアがユニークで，未来の創作文化を予感させるものであるか），完成度（Web アプリケーションが一般的な Web ブラウザで正常に動作するか，実装が技術的に優れているか）であった．3 つの評価軸はコンテストの公式 Web サイトに明記され，参加者はいつでも確認できた．我々は，クリエイティビティとイノベーションに関して，選定過程ですべての応募作品を実行して試すことで評価を行い，さらにソースコードを読むことで完成度の評価を行った．また，入選作品選出後，総合判断では入選には至らなかったが，我々がいずれかの評価軸で注目すべきと判断した作品（注目作品）を追加で数件選定した．

そして，入選作品と注目作品を発表するとともに，入選作品を対象としたオンラインの一般投票を受け付けた．1 週間弱の一般投票の期間が終わってから，投票結果をふまえて主催側で改めて入選作品を対象とした審査を行い，受賞作品（最優秀賞 1 件，優秀賞 3 件）を選定した．受賞作品発表後に，事後アンケートを参加者にオンライン配布しており，その結果は 5.4 節で報告する．

5.1.2 フレームワークの参考資料

我々は，Lyric App Framework を公開する際に，チュートリアルや API リファレンス，スライドの読み上げとデモを交えた説明動画などの参考資料を Web サイト[*2]に掲載した．チュートリアルはフレームワークの概要を紹介するとともに，リリックアプリの開発を始めるための手順を説明する内容であった．さらに，フレームワークの使い方をデモするために 11 件のリリックアプリを開発し，GitHub 上で MIT License のサンプルコードとして公開した[*15]．

我々はこれらのリリックアプリがフレームワークの代表的な機能を活用したものになるよう注意深く設計したが，同時に，サンプルコードとして多くの人々の環境で動作するよう，スマートフォンや PC，タブレット端末など多様な Web ブラウザ上で動作する互換性にも気を配った．そのため，実験的な Web 標準 API（位置測位，加速度センサ，Bluetooth など）は利用していない．

5.2 リリックアプリの 8 種類のカテゴリ

まず，リリックアプリの豊かなデザイン空間を理解するため，我々は 2020–2021 年に開催された 2 回のプログラミング・コンテストへの応募作品を分析した．それぞれ 47 日間と 77 日間にわたって応募期間が設けられ，32 件と 20 件の応募があった．提出された全 52 件のリリックアプリはすべて完成して実行可能な状態であった．各リリックアプリの内容を説明する短い文章を書き，その説明文をもとにしたテーマ分析的な手続きを経てリリックアプリを 8 種類に分類した．応募年ごとの各カテゴリの応募数の傾向などの統計情報は，5.3 節で紹介する．

5.2.1 拡張現実アプリ（extended reality）

このカテゴリは，仮想的な 2D・3D の情景における没入感のある体験に重点を置いている．カメラ入力を活用して，実世界の情景に歌詞駆動の視覚的演出レイヤーを追加するものもある．ユーザは楽曲の世界観の中にいるように感じられる．たとえば，特定の楽曲のために作り込まれた，学校の教室の中を自由に動き回れる作品（**図 5**①）があった．この作品では，教室の中に歌詞のストーリーと関連のある隠し要素が配置されており，これを見つける楽しみ方もできるようになっていた．別の作品では，WebVR API を使って VR ヘッドセットで仮想空間に飛び込めるものや，AR.js を使ってカメラ画像の手前に歌詞がタイミングよく表示される立方体を映し出すものなどがあった．

5.2.2 制作ツール（authoring tool）

このカテゴリは，楽曲と同期した派生コンテンツを制作し，作り込み，場合によっては他の人と共有することまで

[*12] https://magicalmirai.com
[*13] https://x.com/TextAliveJp
[*14] https://gitter.im/textalive-app-api/community
[*15] https://github.com/TextAliveJp

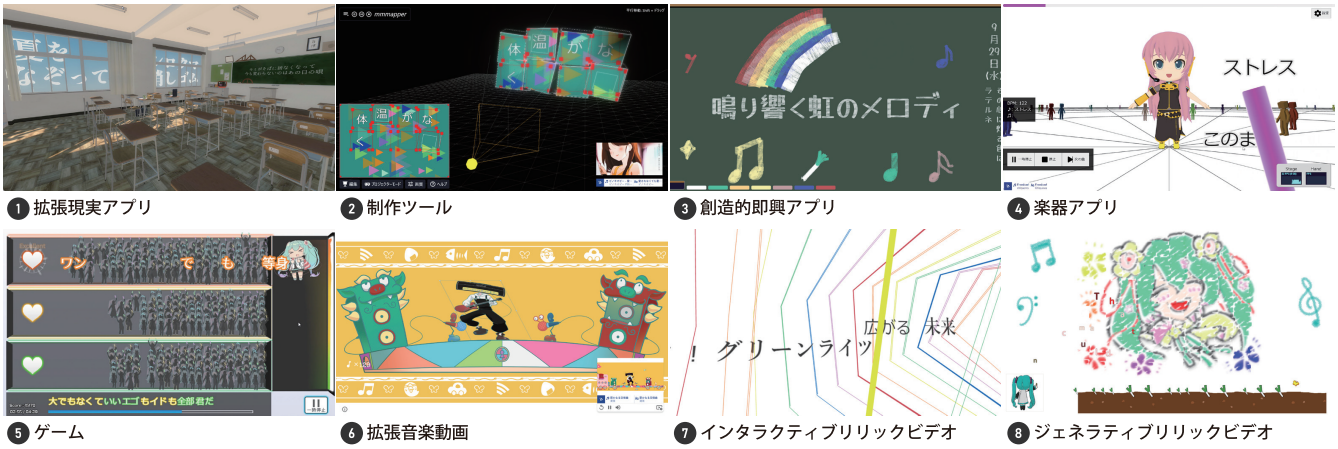

図 5　プログラミング・コンテスト応募作品の分析で判明したリリックアプリのカテゴリ 8 種類と，クリプトン・フューチャー・メディア株式会社提供の代表例のスクリーンショット

Fig. 5　Representative examples of the eight lyric app categories collected in the programming contests, with screenshots courtesy of Crypton Future Media, Inc.

可能にする．たとえば，プロジェクションマッピング用のコンテンツを制作できる作品があった（図 5②）．ユーザは，曲ごとの内容をふまえて定義された仮想的な 3D 空間で，特定の平面上に視覚的要素を配置していくことができた．こうして作り上げた 3D の情景は，実世界で同様の平面を再現すれば実際のプロジェクションマッピングに活用できるようになっていた．

5.2.3　創造的即興アプリ（creative application）

このカテゴリは，ユーザが再生中の楽曲と創造的にインタラクションできるようにしてくれる．制作ツールと異なり，このカテゴリでは基本的にシーク操作が許可されておらず，その代わり即興的な体験に重点が置かれている．たとえば曲を聴きながら曲の終わりまで絵を描ける作品（図 5③）や，タッチ操作で画面上に音符を置いていくとサビでその音符が踊りだす作品などがあった．

5.2.4　楽器アプリ（instrument）

このカテゴリは，ユーザが楽曲演奏に能動的に関与するための楽器や道具の役割を果たす．たとえば，MediaPipe.js を用いたカメラ入力の画像処理によってユーザの手の動きを推定し，画面上の仮想的なペンライトに割り当てた作品があった（図 5④）．また，Web MIDI API を用いて接続された MIDI 機器の信号を読み取り，楽曲再生と同時に演奏ができる作品があった．ユーザのクリック操作のタイミングで既定の音を鳴らす作品もあった．

5.2.5　ゲーム（game）

このカテゴリは何らかのルールや得点制を採用しており，ユーザにゴールしたり高得点をとったりしたいと思わせるような仕掛けが実装されている．典型的なゲームアプリでは，本フレームワークを活用することでユーザが課題曲を選べるようになっている．ゲームに分類された応募作品のすべてが，次々に出現する歌詞テキストとうまくインタラクションすることを目指す音楽リズムゲームであった．たとえば，ユーザがライブステージ上のキャラクタを操作して，飛んでくる歌詞テキストにタイミングよく触れるゲーム作品があった（図 5⑤）．

5.2.6　拡張音楽動画（augmented music video）

このカテゴリは対象楽曲の音楽動画を表示し，その再生と同期したインタラクティブな体験を提供する．たとえば，特定の楽曲の音楽動画が画面右下に表示され，その情景を参考にしたインタラクティブな視覚的要素が画面の他の領域に表示される作品があった（図 5⑥）．キャラクタが画面左から右に歩くシーンでは，ユーザがカーソル操作でキャラクタを左右に動かせるようになっていた．また，操り人形が操られて変形するシーンでは，ユーザがマウス操作で変形の仕方をコントロールできるようになっていた．

5.2.7　インタラクティブリリックビデオ（interactive lyric video）

このカテゴリは，リリックビデオを動的に描画したうえで，ユーザがインタラクションできるようにしたものである．たとえば，歌詞テキストが 3D の仮想空間内を右から左にスクロールする情景を描いた作品では，ユーザがテキストをマウスかタッチのドラッグ操作でつかんで左右に動かし，再生位置を変更できるようになっていた（図 5⑦）．

5.2.8　ジェネラティブリリックビデオ（generative lyric video）

このカテゴリはリリックビデオをプログラムによって動的に描画するもので，単なる動画であればアスペクト比やフレームレートが固定であるのに対し，それらをユーザの閲覧環境に応じて柔軟に変更できるものが多い．シーク操作のような基本的な再生操作は可能だが，それ以外のインタラクションには対応していない．

このカテゴリの典型的なリリックアプリはリリックビデオと似た体験をユーザにもたらすが，好みのスタイルにカスタマイズできる作品があるほか，視覚表現として見たと

きに，歌詞のタイミングやその他の音楽的要素に合わせてアニメーションする同期的演出が多用される特徴がある．なお，リリックビデオでは見られない表現を追求した例外的な作品もあった．たとえば，楽曲全体を特定の数のパートに分割し，イラストも同数のパネルに分割して，それぞれパズルのピースに見立てた作品があった（図 5⑧）．楽曲の再生が進むにつれ，ピースが画面上の適切な場所にはめこまれ，楽曲再生が完了するタイミングでイラストが完成する演出となっていた．

5.3 リリックアプリの特徴に関する追加検証

我々は，2022–2023 年に開催された 2 回のプログラミング・コンテストへの応募作品を追加分析した．いずれも 98 日間にわたって応募期間が設けられ，それぞれ 68 件と 39 件の応募があった．5.1.1 項で紹介したとおり，我々は主催側としての審査の一環で，提出された全 107 件のリリックアプリについて実行し，ソースコードを読んだ．全件が実行可能な状態であったが，とくに多数の応募があった 2022 年に関して，サンプルコードを少しだけ改変したものも散見された．全件を分析対象としてしまうと，それらがすべてインタラクティブリリックビデオに分類されてしまうなど，リリックアプリの特徴に関する理解を深める妨げになることが予想された．そこで，2022–2023 年に関して，審査過程で作品として一定の評価を得たものに絞って追加分析の対象とした．具体的には，各年，入選作品 10 作品と注目作品 3 作品あわせて 13 作品ずつを対象とした．

まず，追加分析対象の各作品がどのカテゴリに分類されるかを集計しながら，分類不能の作品がないか確認した．結果として，分類不能の作品は見られず，2020–2021 年の全件と，2022–2023 年の追加分析対象に関して，カテゴリ別の作品数は**表 2** のとおりとなった．

次に，各カテゴリに分類された作品の内容を検討したところ，2020–2021 年の従来作品とは異質なものが見つかった．これらに関しては，カテゴリの定義を詳細化し，複数のカテゴリに分割することが妥当かもしれない．たとえば，従来の①拡張現実アプリがユーザの没入的な体験に注力したものだったのに対し，自撮りカメラに写ったユーザの口から歌詞があふれ出る作品は，ユーザの身体性に訴えかける点が新しく，人間拡張の要素を含むサブカテゴリに発展する可能性がある．⑦インタラクティブリリックビデオは，他に収まらないインタラクティブ作品の受け皿となっている．たとえば，ソーシャルネットワークのタイムラインを模したインタフェース上で初音ミクがリアルタイムに，他のユーザに混じって歌詞をつぶやいていく作品は，日常にリリックアプリが溶け込んだ未来を予感させる．

表 2 リリックアプリのカテゴリ別作品数
Table 2 Number of lyric apps per category.

	2020	2021	2022	2023	計
①	6	3	1	1	12 (15.4%)
②	4	2	0	0	6 (7.7%)
③	2	4	1	0	7 (9.0%)
④	1	2	0	1	4 (5.1%)
⑤	2	4	3	5	14 (17.9%)
⑥	1	1	0	0	2 (2.6%)
⑦	4	0	5	4	13 (16.7%)
⑧	12	4	3	2	21 (26.9%)
計	32	20	13	13	78

また，それまでのカテゴリの定義そのものであっても，さまざまな面で工夫を重ねたアプリが次々応募されてきている．たとえば，「マジカルミライ」イベント 10 周年だった 2022 年には，楽曲に合わせて 10 年を振り返る⑦インタラクティブリリックビデオが複数あり，インタラクティブなマルチメディアの強みを生かし，複雑な文脈を反映したコンテンツ制作が可能であることが分かった．同カテゴリでは，インタラクションによって演出がリアルタイムに変化したり，展開が分岐したりするゲーム性を帯びた作品や，ユーザの入力した文章を自然言語処理で理解して反応を返す作品など，演出の完成度が高いもの，技術的に高度なものなどが見受けられた．⑤ゲームのカテゴリでは，複数人でのネットワークプレイに対応し，遠隔地でも同時に音楽を聴きながら楽しめるもの，ライブで特定の歌唱部分に対して聴衆が同じタイミングでリアクションを返す「コール＆レスポンス」を練習できるもの，あえてユーザの自由度をスライダー 1 つに制約し，放置しても演出が流れるようにすることで気軽に楽しめるものなど，多くの応用可能性があることが分かった．

このように，リリックアプリをカテゴリに分けたうえで作品内容を検討する定性的検証により，既存カテゴリの分割の可能性が示唆された他，カテゴリ分けとは別の多様な軸での工夫が発見でき，リリックアプリのデザイン空間の質的理解が深まった．一方で，コンテスト形式は毎回の応募数が変動するため，作品数の増減を定量評価することは難しい．内容面でも，サーバサイドのプログラムを含められなかったり，ソーシャルな機能の実装や実験的な API の採用が見送られがちだったりしてきた．また，過去の審査結果が未来の作品応募に影響を与える面もある．今後は，コンテスト以外の方法も併用して，リリックアプリに関する理解を深めていくことが望ましいだろう．

5.4 ユーザ評価

我々はプログラミング・コンテスト参加者に自由記述でのアンケート回答を依頼し，2020 年に 21 件，2021 年に 15 件の回答が寄せられ，コンテストに参加した動機やフレー

ムワークに関するコメントなどの質的フィードバックを収集できた．たとえば，プログラミング・コンテストが創造的な文化にプログラミングの力で貢献する貴重な機会となっていることを大きく評価する旨のコメントがあった．フレームワークに対しては肯定的な感想が大部分を占めたものの，致命的ではないが修正すべきバグに関する報告も寄せられた．

2021 年のアンケートでは参加者のプログラミング経験について詳細に尋ねた．回答によれば，まったくの未経験から 6 年の JavaScript 実務経験まで多様で，フレームワークの敷居が低く，天井が高い（熟達者の創造性を制限しない）ことを示している．とくに，回答者の実に 3 分の 1 が JavaScript を用いたプログラミングの経験がほぼゼロであったことは驚きであり，それでもコンテストに参加した動機について 6.2 節で考察する．

このアンケートでは，参加者が時刻駆動型 API に満足しているか，そして歌詞などの音楽的要素に対するイベントリスナ（`player.addEventListener("phraseEnter", listener)` など）の必要性を感じているかについても質問した．回答は現状の API 設計に肯定的で，リリックアプリの状態管理用に 4.2.1 項のイベント駆動型 API は利用しているものの，我々の提案する時刻駆動型 API は合理的であり，リアルタイムなマルチメディア演出において有用であると考えていた．1 人の回答者のみ，イベントの種類が多いほどよいと回答していたが，他の回答者からはそうした要望はなかった．別の回答者は，複雑なシーン管理を実現するためには現在の API 設計では不十分だとコメントしていた．

6. 議論

本研究は，3 つの幅広い観点で研究上の貢献がある．まず，音楽情報処理の研究者，より広くは音楽業界の人々にとって，音楽コンテンツを取り巻くインタラクティブな体験の重要性は従前より指摘されており [16]，リリックアプリはそうした体験を提供する有望な 1 手段となる．本フレームワークはリリックアプリの開発を支援する初めての取組みであり，実証実験の第 1 の目的であったデザイン空間の探索が実施できたことも含めて，リリックアプリをニッチからメインストリームへ押し上げる重要な技術的・文化的マイルストーンである．

次に，実証実験の第 2 の目的であるフレームワークの有効性検証の結果，6.1 節で後述するとおり，幅広い熟達度のプログラマにとって有用であることが分かっただけでなく，精密な時刻同期が必要なインタラクションデザイン全般で有用な知見が得られた．

最後に，実証実験の第 3 の目的であった，リリックアプリがプログラマの創作コミュニティへの参画機会を生むという仮説は，6.2 節で後述するとおり検証できたといってよく，さらに，創造的文化に関する今後の研究への示唆も得られた．

6.1 時刻駆動型 API に関する知見

5.4 節で紹介したとおり，本フレームワークは幅広い熟達度のプログラマが利用でき，とくに時刻駆動型 API は肯定的に評価された．本節では，さらにコンテスト応募作品のソースコードを読解して得られた知見として，高度な演出をプログラミングするには再利用可能なコンポーネントが重要なことを述べる（6.1.1 項）．さらに，そうしたデザインパターンを支援するための新たな API 設計を提案，実装，公開したことを報告する（6.1.2 項，6.1.3 項）．

これらの知見は，精密な時刻同期が必要なインタラクションデザイン全般に応用できる．たとえば，音と映像を同期させる演出のほか，センサとアクチュエータを活用した実世界インタラクションの開発支援でも有用だと考えられる．

6.1.1 リリックアプリにおける再利用可能コンポーネント

コンテスト応募作品のほとんどが，クリエイティブコーディングや GUI のライブラリのレンダリング用関数（4.2.2 項）を利用して画面演出をプログラミングしていた．このようなジェネラティブなアルゴリズムは，単一の巨大な関数としても実装できるが，こうした方法は単調な映像表現につながりやすく，我々はアンチパターンだと考えている．

我々が分析したソースコードの中では，関数を細分化して多くの再利用可能なコンポーネントとして扱うデザインパターンに沿ったものが，モジュール性を向上できており，複雑で魅力的な画面演出に成功している傾向にあった．

6.1.2 時区間駆動型（time-range-driven）API

リリックアプリ開発において，前述のような再利用可能コンポーネントのライフサイクルが，音楽的要素を表す時区間（ビートから次のビートまでの区間やフレーズの発声区間など）に紐づいていることは注目に値する．我々は，分析したソースコードの中で，現在の再生位置が対象の時区間から出入りしたことを検出する機能がたびたび実装されていることを発見した．たとえば，ある作品では，画面上のパーティクルの出現を管理するパーティクルマネージャが，前回の `player.findPhrase` 関数の呼び出し結果を保存し，前回と今回の結果を比較することで，パーティクルを表すオブジェクトを初期化するか破棄するか決定していた．

このデザインパターンを支援するため，我々は 2024 年 5 月 12 日に TextAlive App API v0.4.0 [*16] を公開し，「時区間駆動型（time-range-driven）API」(図 3ⓒ) を提供した．時刻駆動型 API が単一時刻を与え単一オブジェクトを問い合わせるものであったのに対し，本 API は，次のように開

[*16] https://github.com/TextAliveJp/textalive-app-api/releases/tag/v0.4.0

始時刻と終了時刻からなる左開右閉区間を与えて，その区間に起きたことをまとめて問い合わせられる API である：

```
player.findPhraseChange(previousPosition,
    currentPosition);
```

返値は，その時区間で発声を終えたフレーズ群（`left`），発声を始めたフレーズ群（`entered`），発声中のフレーズ（`current`），そして区間外で近接する前後のフレーズ（`previous`，`next`）である．

時区間駆動型 API は，プログラマが自身のコードで能動的に情報をとりに行く時刻駆動型 API の特長が保たれており，さまざまなクリエイティブコーディングや GUI のライブラリと併用できる．さらに，各ライブラリが提供するレンダリング用の関数内で毎回呼び出すことで，対象時区間に入った音楽的要素と，対象時区間を出た音楽的要素を漏れなくすべて取得できるため，パーティクルマネージャのような仕組みの実装が容易になる利点がある．

6.1.3 ユーザ定義時区間

応募作品の多くが，楽曲全体を複数の時区間に区切って，時区間ごとに歌詞駆動演出を表示するように実装されていた．その際，楽曲再生の進行と演出とのマクロな対応付けを行い，時区間の遷移を管理する「シーンマネージャ」を実装するのが一般的であった．こうした時区間は，演出を組み立てるプログラマが独自に考えつく創造的な努力の成果であり，フレームワーク側ではあらかじめ定義できない．

そこで，プログラマが自身で時区間を定義したうえで再生位置と比較する API を提供できれば，こうしたシーンマネージャの実装を支援できる．これは，5.4 節で複雑なシーン管理の支援が必要と要望されていた機能に他ならない．具体的には，ACM CHI 2023 論文 [4] の議論では `Player` インスタンスを介して利用する密結合の API 設計案を提示した．ただし，プログラマの利便性の観点からは疎結合のほうが使いやすいため，TextAlive App API v0.4.0 で以下のような API を提供した：

```
// 抽象クラス TimedUnit を継承し独自の時区間を定義
class TObj extends TimedUnit { constructor(
    public startTime: number, public endTime:
    number) {super();} }
// 複数の時区間からなるタイムラインを配列として定義
const tl = [new TObj(0, 10), new TObj(20, 30),
     new TObj(30, 50)];
// find*Change と同じ使い勝手で独自のシーン検出が可能
findTimedObjectsInRange(tl, 25, 40);
// 2 区間目がleft 3 区間目が entered と current
```

6.2 静的メディアへのインタラクティビティ付与

リリックアプリ開発は，既存の楽曲にインタラクションの機能を付与するプロセスであり，プログラムを何もない状態からコーディングし始める体験とは異なる．本節では，こうした，既存のコンテンツを拡張するプログラミング体験の利点と，今後の研究に向けた論点を議論する．

6.2.1 創造性のキックスタート

我々は，実証実験を通して，本フレームワークが典型的なクリエイティブコーディングのコミュニティよりも広範なコミュニティに訴求できており，参加者の開発成果公開を促進できていると感じた．ある参加者は，通常であればこうしたコンテストには参加しようと考えないし，参加できなかっただろうとコメントしていた．コンテスト課題曲への愛が，こうした参加者にリリックアプリ開発とコンテスト参加を促していた．クリエイティブコーディングは近年コンピュータ科学教育や芸術的意義の観点で注目を集めているが，すべてのプログラマが，自身が（クリエイティブコーディングを楽しめるほどに）十分「創造的（クリエイティブ）」だという自信を持っているわけではない．

リリックアプリには，そうしたプログラマが「創造的」な部分で既存の楽曲の力を借りて，結果として自身の創造性を発揮するキックスターターにできるポテンシャルがある．我々は，プログラミングという活動それ自体が十分「創造的」であると考えており，より多くの人々が創造性を発揮するきっかけにリリックアプリがなってきたことを嬉しく思っている．こうしたリリックアプリ開発は，コンピュータ科学教育のカリキュラムでも役立つはずである．さらに，音楽以外の静的メディアコンテンツに対してもインタラクティビティを付与するようなクリエイティブコーディングへの発展も，今後の研究として興味深い．

6.2.2 コミュニケーションとしてのプログラミング

リリックアプリの開発過程で，プログラマは何度も繰り返し楽曲を聴き，歌詞を読むことになる．こうした体験を通して，プログラマは魅力的な歌詞駆動型の演出を思いつくのかもしれない．リリックビデオがミュージシャンとデザイナの協力関係を可能にしたことと同様に，リリックアプリはミュージシャンがプログラマと協力する新たな機会を生み出している．「プログラミング環境」はコンパイラやエディタなど単なるソフトウェアの集合と見なされがちだが，実際には人々と人工物（ソフトウェアと，参考資料などのコミュニケーション媒体）が織りなす共創的な環境である [17]．たとえば，拡張音楽動画のリリックアプリ（図 5⑥）を開発した参加者は，元の楽曲と音楽動画の作者に問合せを行い，派生作品の共創的な開発を楽しんでいた．

プログラミング・コンテストの受賞者を発表する「マジカルミライ」の企画展ステージにおいて，コンテストを主催するクリプトン・フューチャー・メディア株式会社の代表取締役であり，音楽業界に詳しい伊藤博之氏は，音楽と技術の密接な関係に以下のように言及した．『ミュージシャンが聴衆に音楽を届ける方法はつねに科学的発明の影響を受けてきた．リリックアプリは，プログラマが創作文化に貢献できる機会を増やす新しいプラットフォームであり，

今後の展開に期待している.』(抜粋のうえ，要約して掲載)

6.2.3 創造的文化に関する今後の研究

創造性支援ツールの研究は長らく，技術的新規性に偏重したものが多く，評価実験もラボスタディなど統制的な環境が主流で，「在野（in the wild)」での研究が不足していると批判されてきた [6]．コンテンツの価値が，派生作品や聴衆の SNS での関与なども含めて総体的に評価されるユーザ参加型文化 [2] の時代において，コンテンツを取り巻く生態系はますます広がりを見せており，その実態をふまえた研究のためにも，在野での研究の重要性は増している．

本フレームワークは一般公開されており，まさに在野での取組みと見なすことができるが，本論文で報告した実証実験でも，コンテスト形式に起因する制約はまだ残っていた（5.3 節）．我々は，本フレームワークの研究開発を続け，コンテストだけでなく，作品を定常的に展示できるプラットフォームの構築などを通して，長期的にリリックアプリの発展を見守り，支えていく必要があると考えている．在野での活用支援を継続することで，新たなリリックアプリのカテゴリが切り拓かれ，創造的文化に関する構成論的研究をさらに進められると信じている．

7. おわりに

本論文では，歌詞駆動型の新たな表現形式「リリックアプリ」を提案し，プログラマがリリックアプリを開発しようとした際に生じる技術的課題を整理したうえで，それを解決するフレームワーク「Lyric App Framework」を提案した．このフレームワークは時刻駆動型 API を備えた TextAlive App API をはじめとする新規技術の組合せにより，プログラマのリリックアプリ開発とミュージシャンなどによるアプリの事後的なカスタマイズを可能とするもので，2020 年 9 月 18 日に一般公開した．そして，プログラミング・コンテストを開催することで，リリックアプリのデザイン空間を探索して 8 つのカテゴリを明らかにし，フレームワークの有効性を検証するとともに新たな時区間駆動型 API を提案・実装し，創作文化のコミュニティと一緒にリリックアプリの可能性を模索してきた．今後もこうした研究を発展させ，多彩な人々が多様な創造性支援ツールにより創造性を発揮する未来像を実現していきたい．

謝辞 プログラミング・コンテストはクリプトン・フューチャー・メディア株式会社が運営した．こうした取組みは，新しい技術が人々の創造性に貢献するという信念を共有した信頼・協力関係のもとで可能となったものである．また，継続的なコンテスト開催は，情熱的な参加者による応募がなければ実現しなかった．参加者の尽力に感謝するとともに，リリックアプリの可能性を切り拓く学術的取組みの中で応募作品を紹介できたことを嬉しく思う．

本フレームワークの Web ベースのワークフローに貢献した産総研の関係者各位にも感謝したい：中野倫靖（歌詞同期エンジン)，渡邉研斗（形態素解析 API)，川崎裕太・井上隆広（Songle および Songle API)．本研究の一部は以下の支援を受けた：JST CREST（JPMJCR20D4)，JST ACCEL（JPMJAC1602)，JST ACT-X（JPMJAX22A3)．

参考文献

[1] Kato, J., Nakano, T. and Goto, M.: TextAlive: Integrated Design Environment for Kinetic Typography, *Proc. 33rd Annual ACM Conference on Human Factors in Computing Systems*, *CHI '15*, pp.3403–3412, Association for Computing Machinery (online), DOI: 10.1145/2702123.2702140 (2015).

[2] Jenkins, H.: *Confronting the challenges of participatory culture: Media education for the 21st century*, The MIT Press (2009).

[3] Levtov, Y.: Algorithmic Music for Mass Consumption and Universal Production, *The Oxford Handbook of Algorithmic Music*, McLean, A. and Dean, R.T. (Eds.), chapter 34, pp.627–644, Oxford University Press (online), DOI: 10.1093/oxfordhb/9780190226992.013.15 (2018).

[4] Kato, J. and Goto, M.: Lyric App Framework: A Web-Based Framework for Developing Interactive Lyric-Driven Musical Applications, *Proc. 2023 CHI Conference on Human Factors in Computing Systems*, *CHI '23*, pp.124:1–124:18, Association for Computing Machinery (online), DOI: 10.1145/3544548.3580931 (2023).

[5] 加藤 淳，中野倫靖，後藤真孝：TextAlive：インタラクティブでプログラマブルな Kinetic Typography 制作環境，第 22 回インタラクティブシステムとソフトウェアに関するワークショップ，WISS '14, 日本ソフトウェア科学会，pp.39–44 (2014).

[6] Frich, J., Biskjaer, M.M. and Dalsgaard, P.: Twenty Years of Creativity Research in Human-Computer Interaction: Current State and Future Directions, *Proc. 2018 Designing Interactive Systems Conference*, *DIS '18*, pp.1235–1257, Association for Computing Machinery (online), DOI: 10.1145/3196709.3196732 (2018).

[7] Fujihara, H., Goto, M., Ogata, J. and Okuno, H.G.: 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 (online), DOI: 10.1109/jstsp.2011.2159577 (2011).

[8] 株式会社シンクパワー：歌詞データ関連事業「01. 同期歌詞（プチリリ機能)」(2023)，入手先 ⟨https://syncpower.jp⟩.

[9] Musixmatch s.p.a: Musixmatch Developer API (2022), available from ⟨https://developer.musixmatch.com/⟩ .

[10] LyricFind Inc.: Products | Lyric Display — LyricFind (2023), available from ⟨https://www.lyricfind.com/products/lyric-display⟩.

[11] Reas, C. and Fry, B.: *Processing: A Programming Handbook for Visual Designers and Artists, 2nd Edition*, The MIT Press (2014).

[12] Kato, J., Ogata, M., Inoue, T. and Goto, M.: Songle Sync: A Large-Scale Web-Based Platform for Controlling Various Devices in Synchronization with Music, *Proc. 26th ACM International Conference on Multimedia*, *MM '18*, pp.1697–1705, Association for Computing Machinery (online), DOI: 10.1145/3240508.3240619 (2018).

[13] Goto, M., Yoshii, K. and Nakano, T.: Songle Widget:

Making Animation and Physical Devices Synchronized with Music Videos on the Web, *2015 IEEE International Symposium on Multimedia* (*ISM*), pp.85–88 (online), DOI: 10.1109/ISM.2015.64 (2015).
[14] Goto, M., Yoshii, K., Fujihara, H., Mauch, M. and Nakano, T.: Songle: A Web Service for Active Music Listening Improved by User Contributions, *Proc. 12th International Society for Music Information Retrieval Conference*, *ISMIR '11*, pp.311–316 (2011).
[15] 後藤真孝，吉井和佳，藤原弘将，Mauch, M.，中野倫靖：Songle：音楽音響信号理解技術とユーザによる誤り訂正に基づく能動的音楽鑑賞サービス，情処学論，Vol.54, No.4, pp.1363–1372 (2013).
[16] Goto, M. and Dannenberg, R.B.: Music Interfaces Based on Automatic Music Signal Analysis: New Ways to Create and Listen to Music, *IEEE Signal Processing Magazine*, Vol.36, No.1, pp.74–81 (online), DOI: 10.1109/MSP.2018.2874360 (2019).
[17] Kato, J. and Shimakage, K.: Rethinking Programming "Environment": Technical and Social Environment Design toward Convivial Computing, *Conference Companion of the 4th International Conference on Art, Science, and Engineering of Programming*, *Programming '20*, pp.149–157, Association for Computing Machinery (online), DOI: 10.1145/3397537.3397544 (2020).

推薦文

登壇発表部門の査読において研究内容が高く評価されて，最優秀論文賞に選出されたため．

（インタラクション 2024 プログラム委員長　水野慎士）

加藤 淳　（正会員）

2014 年東京大学大学院情報理工学系研究科博士後期課程修了．博士（情報理工学）．在学中 Microsoft Research（Asia/Redmond），Adobe Research Seattle 研究インターン．2014 年国立研究開発法人産業技術総合研究所研究員，2018 年同主任研究員．同年アーチ株式会社技術顧問を兼務．2024 年度 Universite Paris-Saclay で在外研究．ACM CHI 2013/2015/2023 Honorable Mention Award，2021 年 IPSJ/ACM Award，2023 年 Japan ACM SIGCHI Chapter 優秀若手研究者賞等 22 件受賞．

後藤 真孝　（正会員）

1998 年早稲田大学大学院理工学研究科博士後期課程修了．博士（工学）．現在，産業技術総合研究所首席研究員．2009〜2017 年に IPA 未踏 IT 人材発掘・育成事業 PM，2016〜2022 年に JST ACT-I「情報と未来」研究総括を兼任．現在，JST 創発 PO，日本学術会議連携会員，統計数理研究所客員教授，筑波大学連携大学院教授等を兼任．日本学士院学術奨励賞，日本学術振興会賞，ドコモ・モバイル・サイエンス賞基礎科学部門優秀賞，市村学術賞，FIT 船井業績賞，星雲賞等，69 件受賞．本会フェロー．