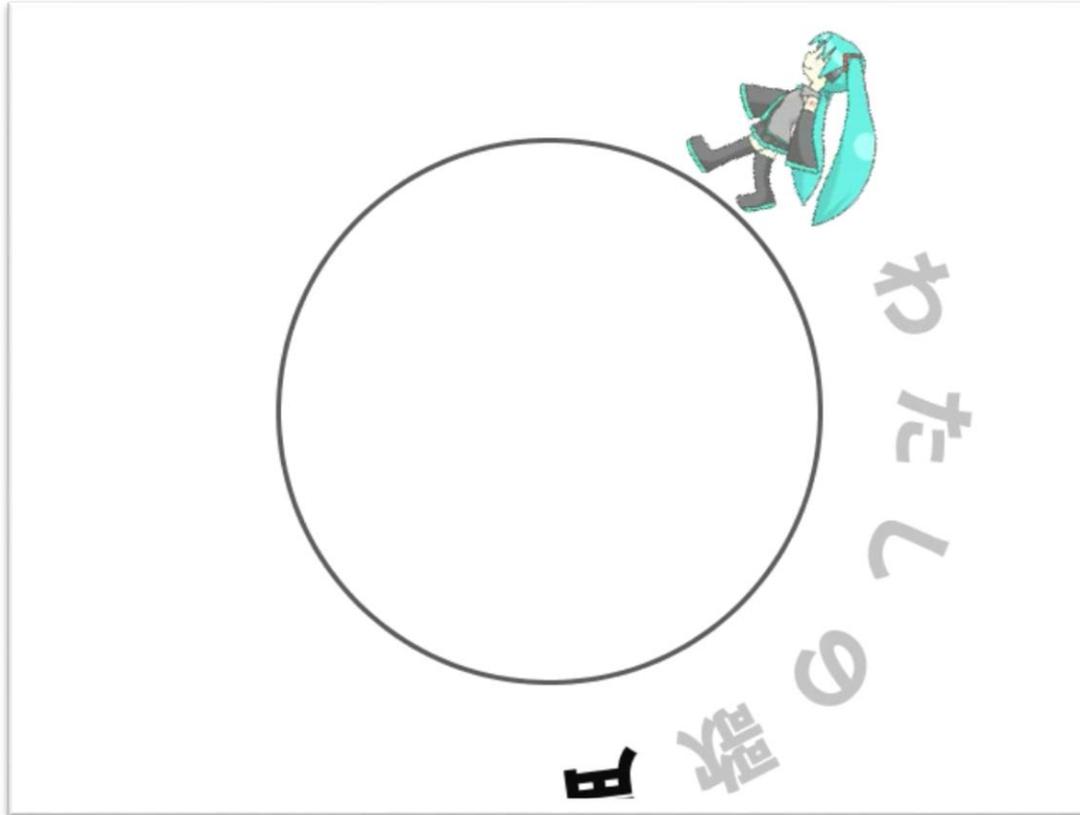


# TextAlive: 音楽に同期した 歌詞アニメーションの Kinetic Typography制作環境

加藤 淳 中野 倫靖 後藤 真孝  
産業技術総合研究所

# Kinetic Typographyとは



# Kinetic Typographyとは

静止したテキストよりも、文字情報を印象的に見せることができる**映像表現の一手法**

- 歌詞表示に限らず、テレビ番組のテロップや映画などで多用されている
- 研究上はコミュニケーションツールへの応用が多い



カラオケ表示

[Lee, 2006] [Forlizzi, 2003] [水口, 2005] [Strapparava, 2006]など

# 音楽の楽しみ方の進化

蓄音機

カラオケ

ラジオ

音メディアの時代

ビデオ

ビデオカラオケ

テレビ

映像メディアの時代

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



印刷物上の文字が動画になり、Kinetic Typographyが生まれた

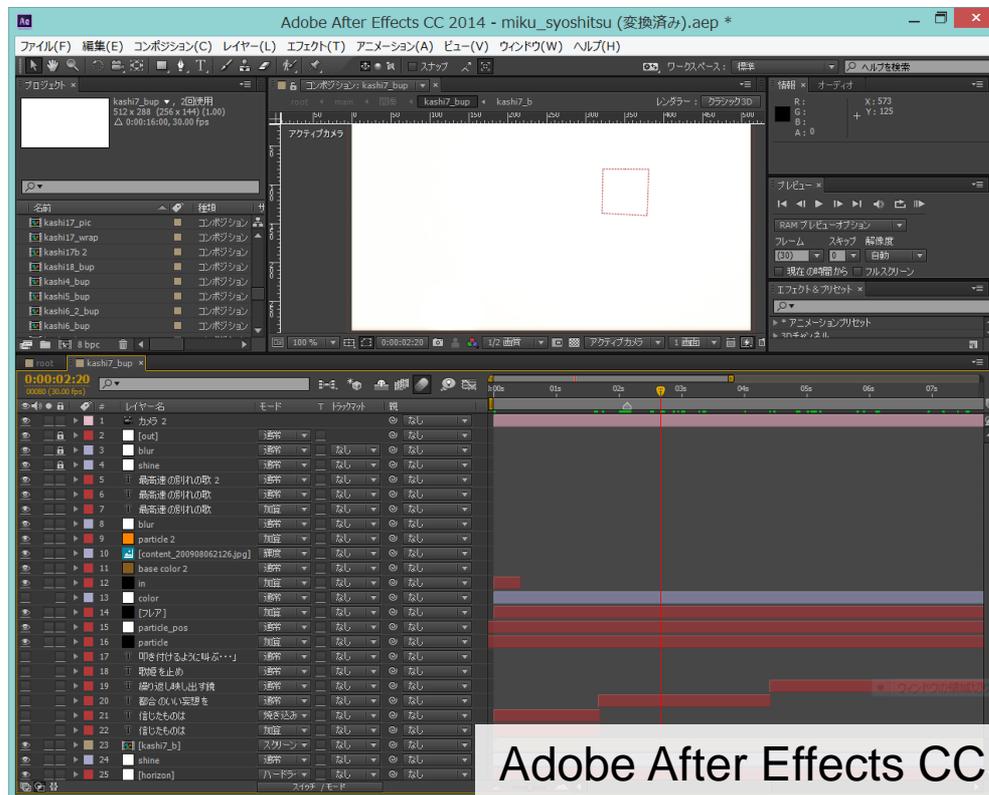
歌詞に込められた物語や想いをより伝えられる  
**Kinetic Typography**動画の重要性が増している

# 一般的なKinetic Typography制作環境

- 汎用動画制作ツール
- **Typography**要素を一文字ずつ指示  
例) 字形、サイズ、色...
- **動き**をゼロから設計  
例) 出現タイミング、軌跡...



膨大な手作業



# 歌詞アニメーションの難しさと提案手法①

従来は、一文字ずつタイミングを合わせるしかなかった  
いつ動き始め、動き終わり、どこで目立たせるかetc.



TextAliveでは…

**タイミングが自動的に推定され、修正も容易**

この世界のメロディー

# 歌詞アニメーションの難しさ と 提案手法②

従来は、一文字ずつ動きを設計するしかなかった

どのような軌跡で動かし、他の字とどういう位置関係にするかetc.

歌詞は、フレーズ、単語、文字という階層構造を持つ



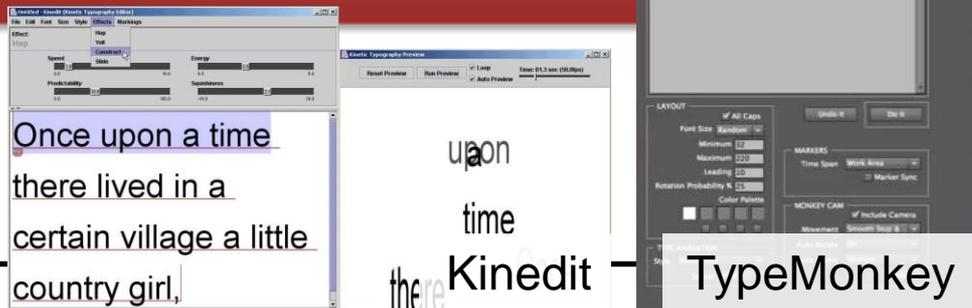
TextAliveでは…

階層毎の動き(テンプレート)を組み合わせて  
複雑な動きを設計可能

# 歌詞アニメーションの難しさと提案手法③

従来、テンプレートは容易に編集できなかった

どのテンプレートも画一的なインタフェースで調整していた



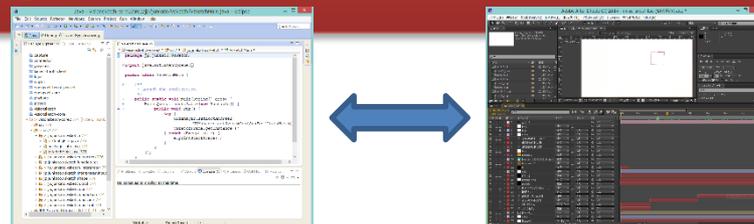
TextAliveでは…

テンプレートごとに独自のユーザインタフェースを  
持ち、見た目を容易に調整可能

# 歌詞アニメーションの難しさと提案手法③

従来、テンプレートは新しく作れなかった

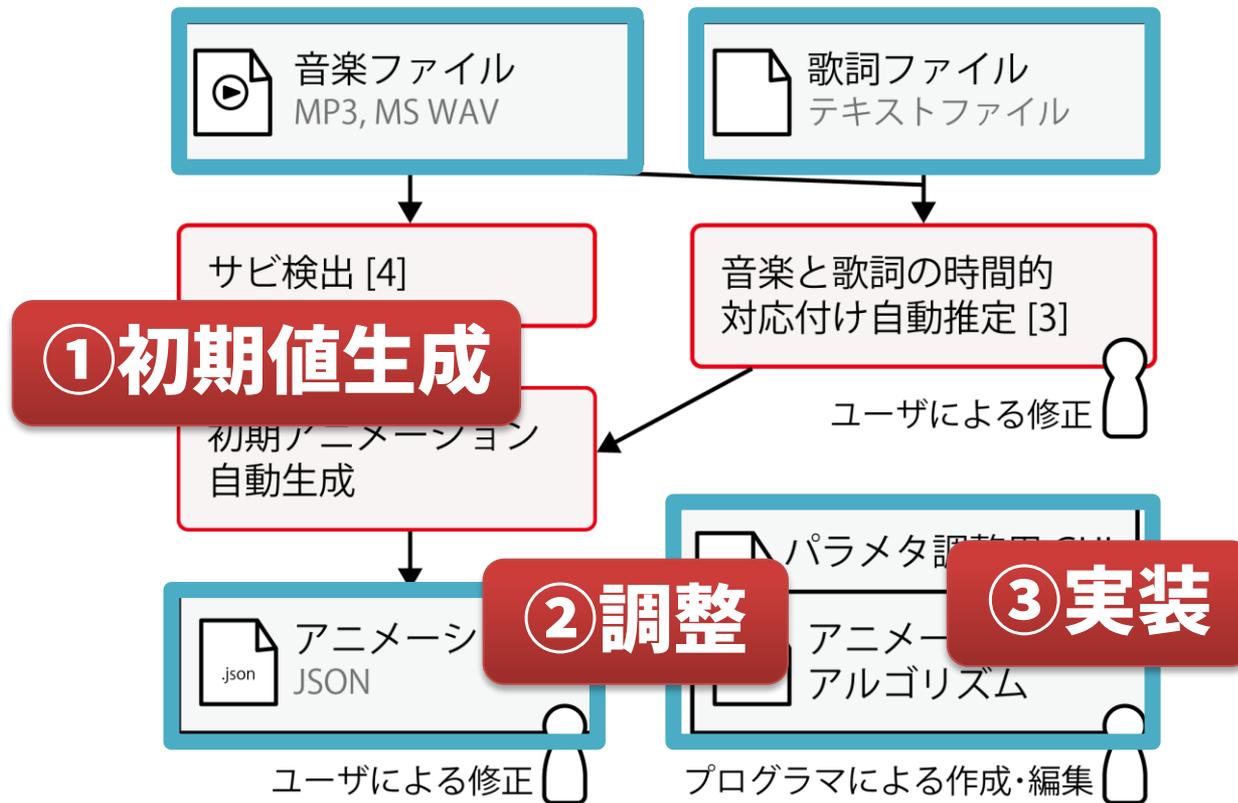
テンプレート制作と動画制作は完全に分離していた



TextAliveでは…

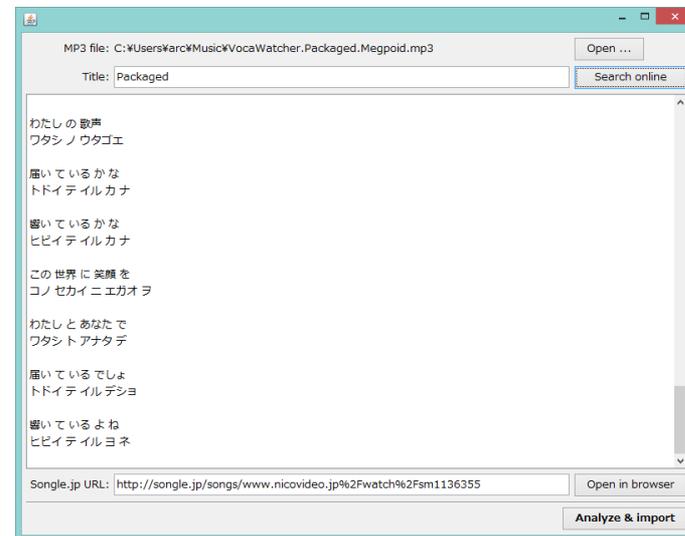
**テンプレートを実装するためのエディタを内蔵し、  
テンプレート・動画制作をスムーズに行き来できる**

# TextAliveの実装



# ①初期アニメーションの自動生成

- 混合音中の歌声と歌詞の自動同期 [藤原, 2011]
- サビの自動検出 [後藤, 2006]
- タイミング情報をもとに、以下の割り振りを自動で行う
  - サビは派手なアニメーション
  - 他はおとなしいアニメーション



TextAliveの実装

## ②テンプレートのパラメタ調整用UI

- テンプレートのソースコードに1行書き足すと表示される

@ui Slider(最小値, 最大値)

offset: 2339



@ui Radio(項目名1, 項目値1, ...)

horizontalAlignment:

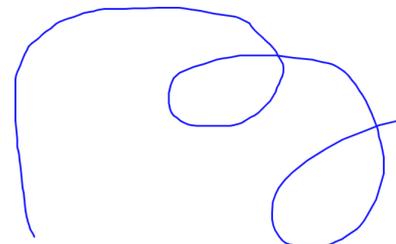
left  center  right

@ui File()

imagePath

@ui Track()

Draw track



# ③ テンプレートのLive Programming

- テンプレートはJavaクラスとして実装
- publicフィールドに状態情報を持っていると仮定

「Save and update」ボタンを押した時の処理 (概要):

編集集中のテンプレートを利用している  
アニメーションのシリアル化 (JSON化)

テンプレートインスタンスおよびクラス情報の破棄

新クラスのコンパイルとクラスローダーによる読み込み

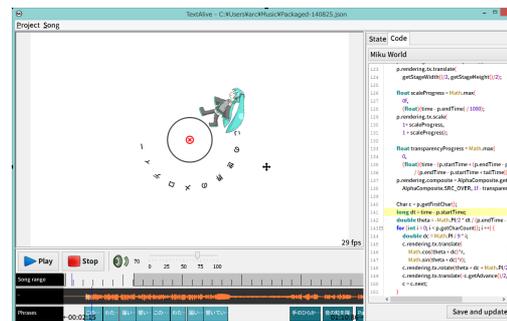
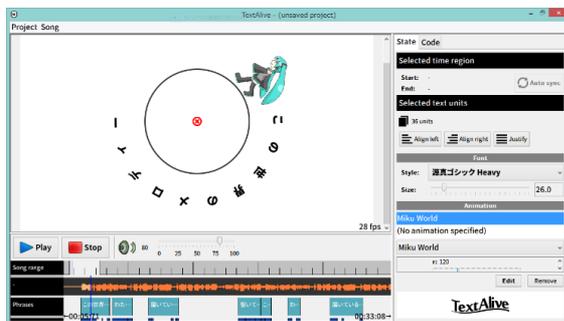
アニメーションをJSON表現からデシリアル化  
(状態情報の復元はベストエフォート)

```
State Code
MikuWorld
123 p.rendering.tx.translate(
124     getWidth()/2, getStageHeight()/2);
125
126 float scaleProgress = Math.max(
127     0f,
128     (float)(time - p.endTime) / 1000);
129 p.rendering.tx.scale(
130     1 + scaleProgress,
131     1 + scaleProgress);
132
133 float transparencyProgress = Math.max(
134     0,
135     (float)(time - (p.startTime + (p.endTime - p
136     / (p.endTime - p.startTime + tailTime))
137 p.rendering.composite = AlphaComposite.get
138     AlphaComposite.SRC_OVER, If - transparer
139
140 Char c = p.getFirstChar();
141 long dt = time - p.startTime;
142 double theta = -Math.PI/2 * dt / (p.endTime -
143 for (int i = 0; i < p.getCharCount(); i++) {
144     double dc = Math.PI / 9 * i;
145     c.rendering.tx.translate(
146         Math.cos(theta + dc)*r,
147         Math.sin(theta + dc)*r);
148     c.rendering.tx.rotate(theta + dc + Math.PI/2
149     c.rendering.tx.translate(-c.getAdvance()/2,
150     c = c.next;
151 }
```

# おわりに

歌詞のKinetic Typography動画制作に関する問題を解決する**統合制作環境TextAlive**を提案した

- 音楽と歌詞を与えると**初期アニメーション**を自動生成
- 好みのアニメーションへ**容易にカスタマイズ可能**
- テンプレートを**Live Programming**できるエディタ



# 今後の予定

- Web上のサービスとして**公開予定**
- テンプレートで利用できる機能の拡充
  - 字形のより自由な変形、3D化…
- 歌詞以外(講演書き起こしなど)への応用
- 評価実験

# TextAlive: 歌詞Kinetic Typographyの制作環境

歌詞のKinetic Typography動画制作に関する問題を解決する統合制作環境**TextAlive**をデモします！

