

TextAlive: Integrated Design Environment for Kinetic Typography

Jun Kato, Tomoyasu Nakano, Masataka Goto
AIST, Japan

Kinetic Typography

この世界のメロディー

Kinetic Typography

- A style of graphical expression to **move and reshape text** in videos
- Prior work focuses on **graphical** aspects, most augmenting text-based communication

[Lee, 2006] [Forlizzi, 2003] [Mizuguchi, 2005] [Strapparava, 2006]

This work focuses on

Animating **lyric text** in synchrony with **music**



Synchronization is difficult & takes long time

Authoring with **specialized tools**

☹️ Set **typographic parameters per character**

e.g. Font, Size, Color, ...

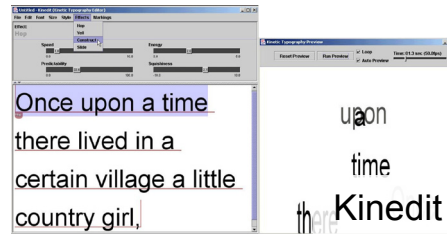
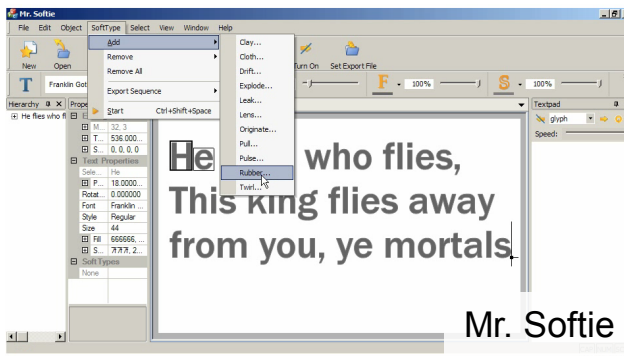
☹️ Specify **kinetic behavior from scratch**

e.g. Timing, Motion path, ...

😊 Choose motion from **templates** and tweak parameters

Authoring with **specialized tools**

☺ Choose motion from **templates**



☹ **Manual labor remains** ☹ **Lack of flexibility**
to specify lyric timings

should be more **effortless** and **flexible**

TextAlive

Problem

Manual effort

Lack of flexibility



Proposed Method



**Automatic Video
Composition**

**Live Programming
of Templates**

For designers

**Interactive
Editing**

For programmers

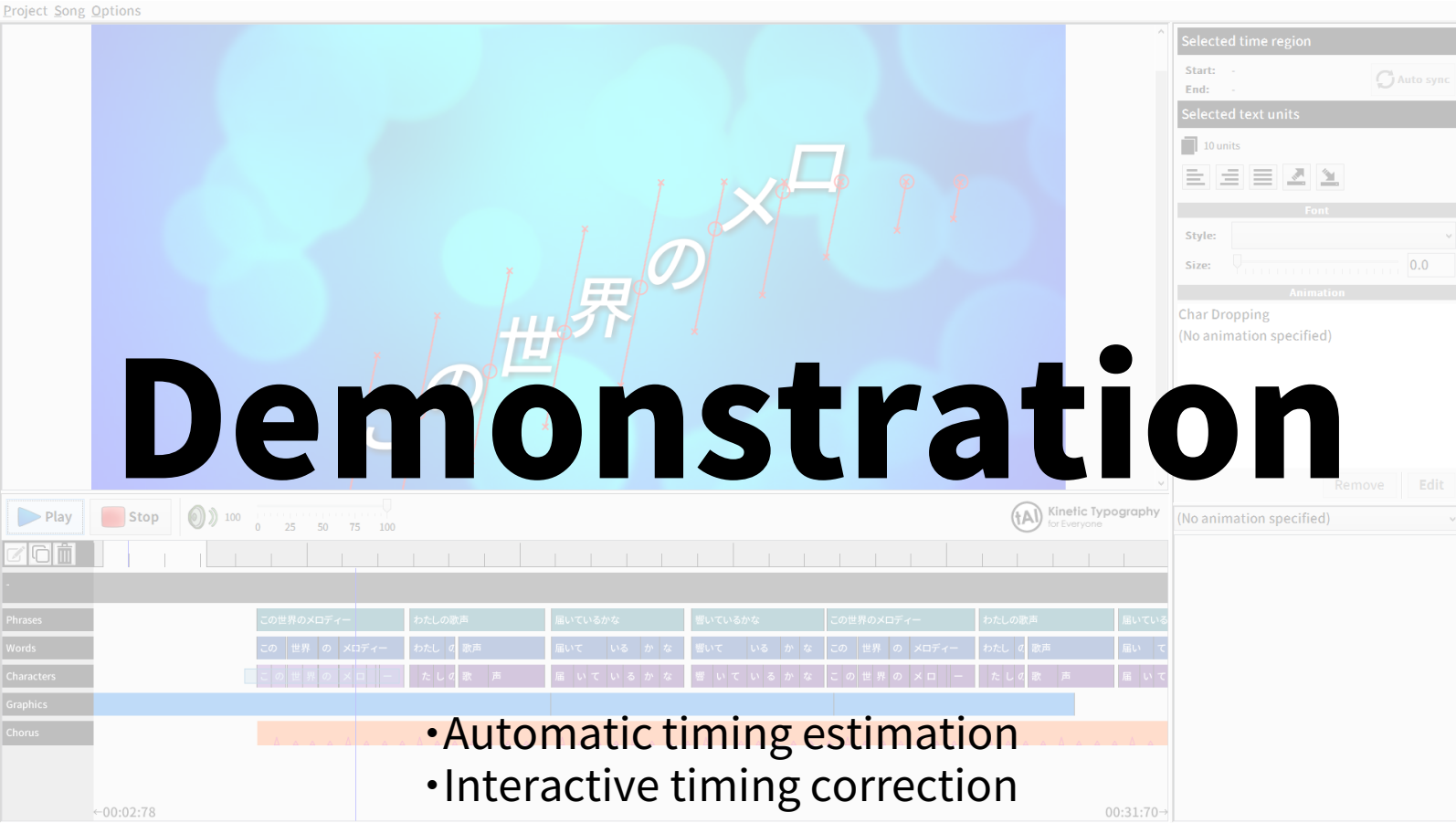
Integrated Design Environment

TextAlive:

Integrated Design Environment for Kinetic Typography

AUTOMATIC VIDEO COMPOSITION + INTERACTIVE EDITING

Project Song Options



Demonstration

• Automatic timing estimation
• Interactive timing correction

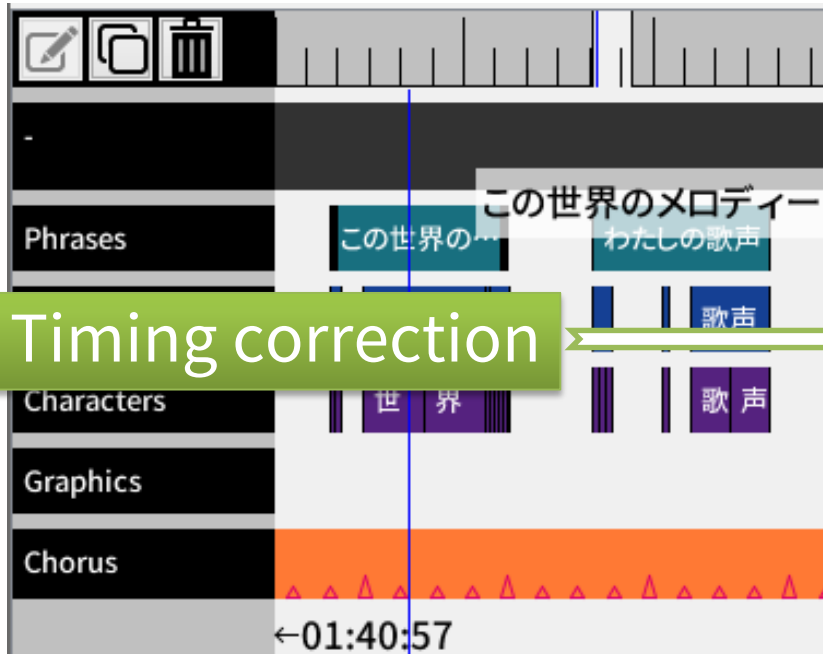
00:02:78 00:31:70

Automatic timing estimation

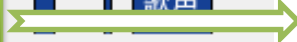


drastically reduces the manual effort

Interactive timing correction



Timing correction



Interactive update



Estimation errors can be easily corrected

Project Song Options



Demonstration

• Structure-aware editing

Phrases	この世界のメロディー	わたしの歌声	届いているかな	響いているかな	この世界のメロディー	わたしの歌声	届いている
Words	この 世界 の メロディー	わたし の 歌声	届いて いる か な	響いて いる か な	この 世界 の メロディー	わたし の 歌声	届いて
Characters	こ の 世 界 の メ ロ -	た し の 歌 声	届 い て い る か な	響 い て い る か な	こ の 世 界 の メ ロ -	た し の 歌 声	届 い て
Graphics							
Chorus							

←00:02:78 00:31:70→

Structure-aware editing

Sliding

この世界のメロディー

Sliding + Hopping

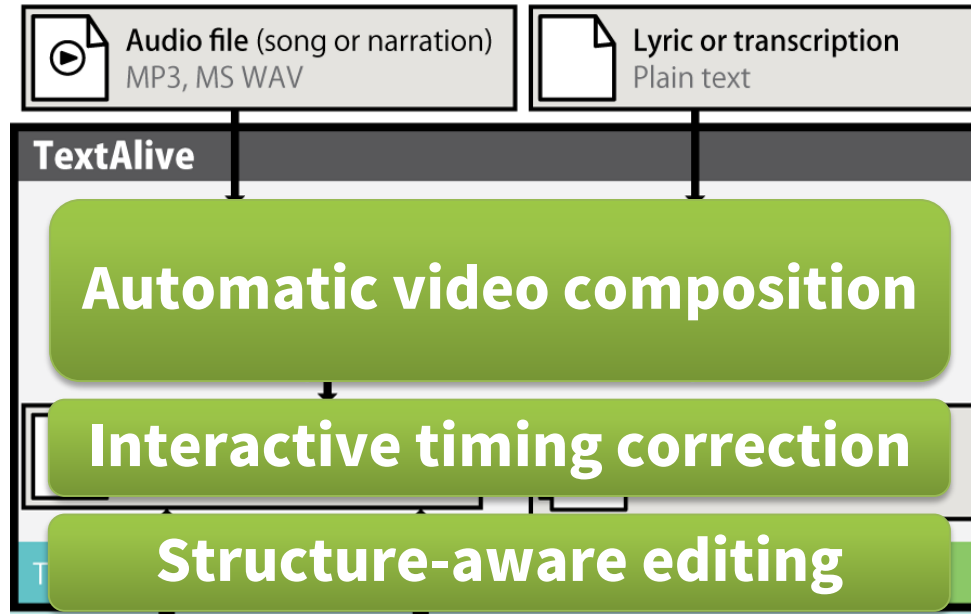
この世界のメロディー

Sliding + Hopping + Karaoke

この世界のメロディー

Multiple templates can be assigned to
phrases/words/characters

TextAlive workflow (for designers)



→ : Data flow or dependency ↔ : User interaction

TextAlive workflow (for designers)



→ : Data flow or dependency ↔ : User interaction

TextAlive:

Integrated Design Environment for Kinetic Typography

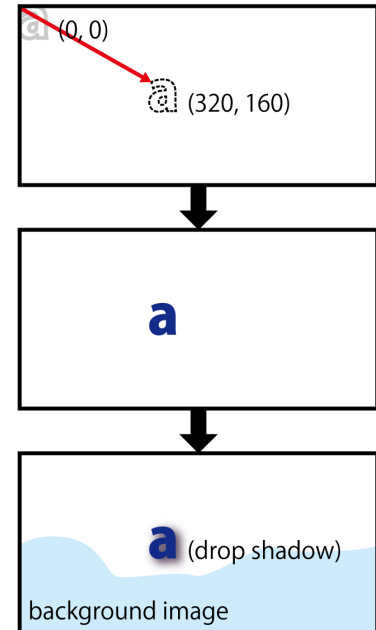
LIVE PROGRAMMING OF ANIMATION TEMPLATES

Video = $f(\text{time})$ without side effects

1. for (all templates)
 - manipulate rendering parameters of assigned text units

```
35 // Make this char hop!  
36 float progress = Math.min((float)(time - c.startTime)/hopTime, 1f);  
37 c.rendering.tx.translate(  
38     0, (float) -hopHeight * Math.sin(progress*Math.PI));
```

2. for (phrases) transform the coordinate system
 - for (words) transform the coordinate system
 - for (characters) transform the coordinate systems;
 - render the character.
3. for (graphic units) render graphics



Project Song Options



Demonstration

• Live programming of animation templates

Phrases	この世界のメロディー	わたしの歌声	届いているかな	響いているかな	この世界のメロディー	わたしの歌声	届いている
Words	この世界のメロディー	わたしの歌声	届いて いる かな	響いて いる かな	この世界のメロディー	わたしの歌声	届いて
Characters	この世界のメロ	わたしの歌	届 いて いる かな	響 いて いる かな	この世界のメロ	わたしの歌	届 いて
Graphics							
Chorus							

←00:02:78 00:31:70→

Live Programming

Project Song Options



Video authoring

Template development

Code Editor: CharDropAnimation

```
40  
41 public void animate(long time) {  
42  
43     // Hide text units if not vocalized.  
44     Char c = (Char) getAssignedUnit();  
45     if (time < c.startTime  
46         || (useAbsoluteTime  
47             ? time > c.endTime + tailTime * 10  
48             : time > c.endTime)) {  
49         c.rendering.setComposite(null);  
50         return;  
51     }  
52
```

```
58  
59     c.rendering.tx.translate(  
60         0,  
61         (float)(isInverted?1:-1)*hopHeight * Mat  
62
```

Save

Close

Char Dropping

☒ useAbsoluteTime

Smooth transition
between **authoring** and **programming**

Dynamic generation of GUI widgets

```
21 | /**  
22 |  * @ui Slider(0, 1000)  
23 |  */  
24 | public int hopTime = 200;
```



offset: 2339



Comments to **public fields** generates GUI widgets bound to the state information

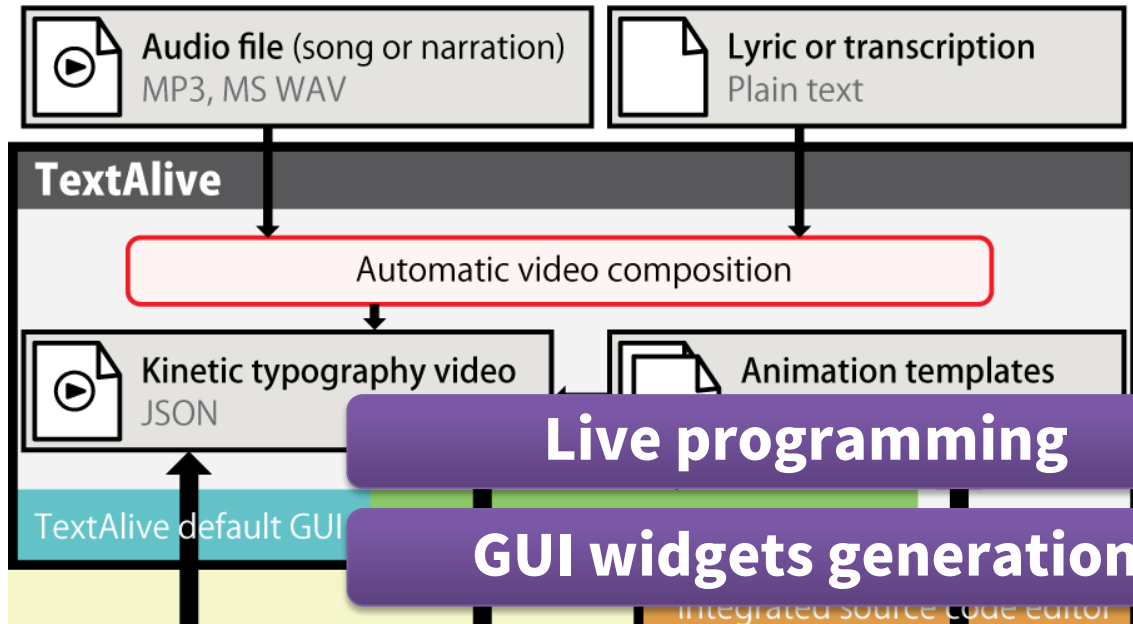
for programmers:

enables easy template **debugging**

for designers:

enables interactive **parameter tuning**

TextAlive workflow (for programmers)



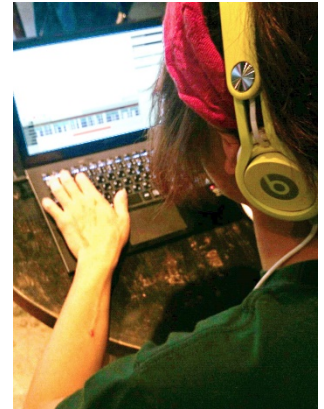
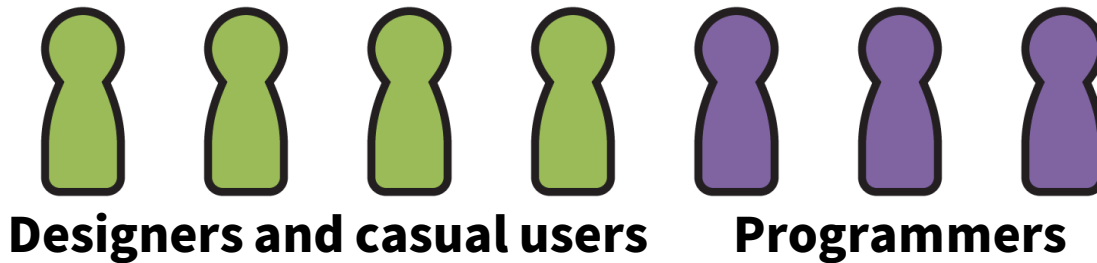
→ : Data flow or dependency ↔ : User interaction

TextAlive:

Integrated Design Environment for Kinetic Typography

PRELIMINARY USER STUDY

Experimental setup

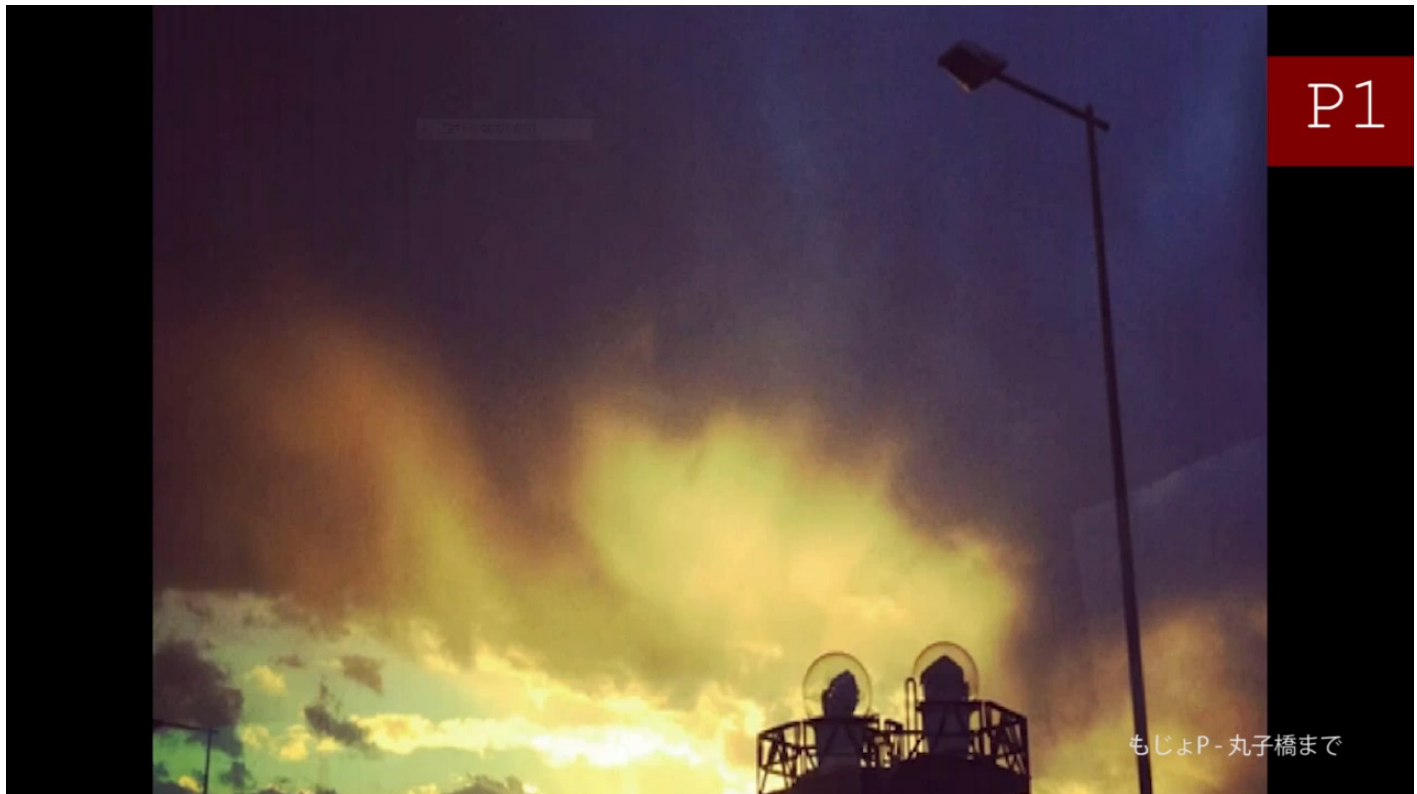


Amateur singer, illustrator, disk jockey, interaction designer

3 have lots of experience, **2** have a little bit of experience,
2 have no experience of using video authoring tools.

Each participant was asked to create a video

Results



Lessons learned: good side 😊

- Automatic video composition is nice since *“we no longer need to create but just edit the video.”*
- Animation templates suggest potential applications of real-time video creation such as *“text jockey.”*
- No need to stop the video = fluid experience
- “Timeline” at the heart of audio-video synchronization

especially for programmers

Integrated interfaces worked as expected

(this should be of interest if you make something similar)

Lessons learned: future work ☹

- Having a variety of animation templates is good
- “Stage” should allow more direct manipulation
- (Programming needs training)

Collaboration between designers and programmers seems to be the key

(and should be an interesting follow-up work)

TextAlive:

Integrated Design Environment for Kinetic Typography

CONCLUSION

TextAlive



The screenshot displays the TextAlive software interface, which is divided into several panels. On the left, a 'Project Song Options' panel shows a video composition window with the Japanese text 'この世界のメロディー' (Kono yo no merodii) overlaid on a blue background. The text is animated with red lines indicating movement. To the right of the video window is a settings panel with sections for 'Selected time region' (Start, End, Auto sync), 'Selected text units' (10 units), 'Font' (Style, Size), and 'Animation' (Char Dropping, No animation specified). Further right is a 'Code Editor: ShakeAnimation' panel showing Java code for an animation class. The code includes package declarations, imports, and a public class definition for ShakeAnimation that extends AbstractAnimation. The class contains two methods, @ui Slider(0, 100), which are commented out. The interface also features three overlapping speech bubbles: a green one on the left saying 'Automatic Video Composition For designers', a red one in the center saying 'Interactive Editing', and a purple one on the right saying 'Live Programming of Templates For programmers'. At the bottom, a large black banner with white text reads 'Integrated Design Environment'.

Automatic Video Composition

For designers

Interactive Editing

Live Programming of Templates

For programmers

Integrated Design Environment

TextAlive

(completely rewritten in JavaScript!)



<http://textalive.jp>

Automatic Video
Composition

For designers

Interactive
Editing

Live Programming
of Templates

For programmers

Integrated Design Environment