Vol.2011-EC-19 No.4
2011/3/26

おどりたい：拍記述により音楽との同期を表現する文字アニメーション記述言語

古屋 啓介[†1] 山本 景子[†1] 倉本 到[†1]
辻野 嘉宏[†1] 水口 充[†2]

現在，テレビ CM などで文字アニメーション表現が頻繁に利用されている．このような文字アニメーション表現を用いる映像の編集には，多くの時間を要するという問題がある．その原因の 1 つとして，映像に用いられる音楽と文字アニメーションのタイミングを合わせるのに手間がかかることが挙げられる．本研究ではその点に着目し，音楽が持つ拍という時間の単位を用いて音楽との同期を表現する文字アニメーション記述言語「おどりたい」を設計，評価した．その結果，全ての文字アニメーションが同じ拍数で動く映像を作る場合,「おどりたい」を用いると既存の言語よりも早く，音楽とタイミングの合った映像を作成できることが分かった．

Beat based kinetic typography markup language to synchronize animated text with music easily

Keisuke Furuya,[†1] Keiko Yamamoto,[†1]
Itaru Kuramoto,[†1] Yoshihiro Tsujino[†1]
and Mitsuru Minakuchi[†2]

Kinetic typography is a text visualization method that changes shape or location of text dynamically, and has been used to add emotive content to film. However, editing these contents require enormous time to synchronize animated text with the music. We focus on beat which is one of the important component of music, and propose a beat based kinetic typography markup language, named "Odoritai", which allows to edit animated texts with music quickly. As a result of experimental evaluation, the proposed language can support editing of animated texts more quickly than the traditional languages especially when the beat patterns of the animation are the same throughout the contents.

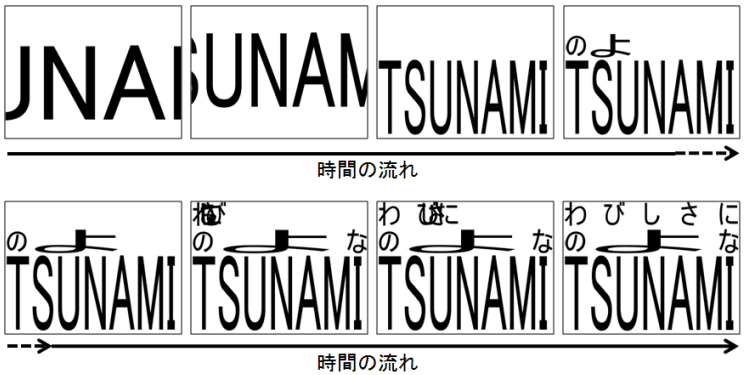

図 1 文字アニメーションの例
Fig. 1 An example of kinetic typography.

1. はじめに

文字アニメーションとは，文字が動いたり文字の色が変わったりする表現をいい，動きによって人の目を引きつけることができる（**図 1**）．そのため，テレビ CM やウェブ上のバナー広告で利用されている．また，文字が動くことで文字が表現する感情を強めることができる．その性質を利用して，音楽のプロモーションビデオなどの芸術作品における表現方法の 1 つとして文字アニメーションが用いられている．動画投稿サイトにおいても，文字アニメーションの持つ見た目の楽しさを利用した動画が見られる．以降，文字アニメーションを用いた映像のことを " 文字アニメーション映像 " と呼ぶ．

現在，文字アニメーション映像を作成するための環境として，文字アニメーション記述言語が開発されている[1),2)]．文献 1) では，プレーンテキストに感情語を割り当てることで文字アニメーション映像を表現する Kinetic Typography Markup Language（KTML）という文字アニメーション記述言語を提案している．文献 2) では，プレーンテキストから Kinetic Typoragphy Description Language（KTDL）という文字アニメーション記述言語を介し

†1 京都工芸繊維大学
Kyoto Institute of Technology
†2 京都産業大学
Kyoto Sangyo University

て，様々な形式の文字アニメーション映像を生成する手法を提案している．

YouTube[4] などでは音楽に合わせて文字が動く文字アニメーション映像を多く見ることができる．このような映像を作る場合，音楽と文字変化のタイミングを合わせなければならない．しかし，既存の文字アニメーション記述言語で文字アニメーションと音楽のタイミングを合わせる場合，音が始まる時刻と音の続く時間を計測して記述する作業が必要となる．特に，音の 1 つ 1 つに文字アニメーションを対応させる場合はこの計測作業に多くの時間を費やさなければならない．一方，通常の音楽は拍に基づいて進行するため，音楽に合わせた文字アニメーション映像を作るときに拍数を用いることで，時間を計測する手間を省くことができる．そこで本研究では，拍に基づいた文字アニメーション記述言語を提案する．また，この記述言語で書かれた文字アニメーション映像を表示するソフトウェアを実現し，その効果を検証する．

2. 音楽に合わせた文字アニメーション映像作成の問題点

文字アニメーション映像の多くは音楽に合わせて文字が動く．例えば[4] では，曲の歌詞が音楽に合わせて動く表現を用いたプロモーションビデオを見ることができる．このような音楽とタイミングの合った文字アニメーション映像を KTDL で作る場合，テキストが表示される時刻，アニメーションの開始時刻および持続時間をそれぞれ指定する必要がある．テキストが表示される時刻とは，文字アニメーション映像中の特定のテキストが表示される時刻である．また，文字アニメーションの開始時刻とは，文字アニメーション映像中の特定のテキストが動き始める時刻のことであり，文字アニメーションの持続時間とは，その文字が動き始めてから動き終わるまでの時間のことである．このため，KTDL では音楽と文字アニメーションのタイミングを合わせる際に，合わせたい音の開始時刻と持続時間を計測する必要がある．開始時刻とは，曲中のある特定の音が鳴り始める時刻のことであり，持続時間とは，その音の長さのことである．

具体的には，ユーザは次の手順で音楽と文字アニメーションのタイミングを合わせる．

1. 波形表示ソフトウェアを用いて音の強さを表す波形を見たり（**図 2**（a)），再生時間を表示しながら曲をスロー再生したりして音の開始時刻を計測する（図 2（b））
2. 同様の手順で音が消える時刻または次の音の開始時刻を計測する
3. 1. と 2. の差を持続時間とし，開始時刻と持続時間を文字の表示される時刻，アニメーションの開始時刻と持続時間に指定する（**図 3**）

この 1. ～3. の手順を，タイミングを合わせたい音全てに対して実行しなければならない．

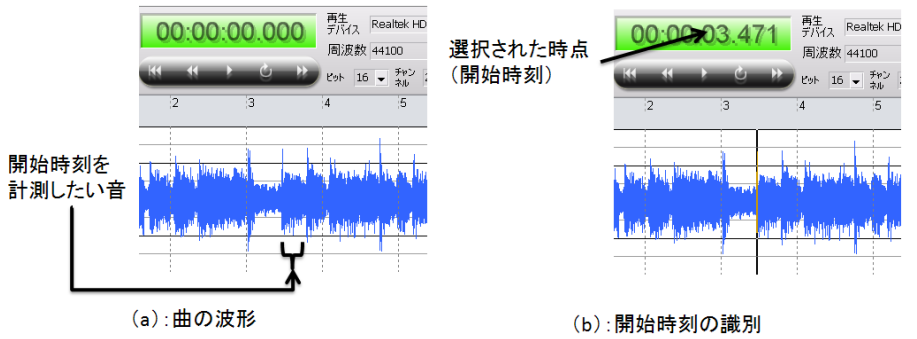

図 2 波形表示ソフトウェアを用いた開始時刻の計測
Fig. 2 Measuring beginning time by using sound editor.

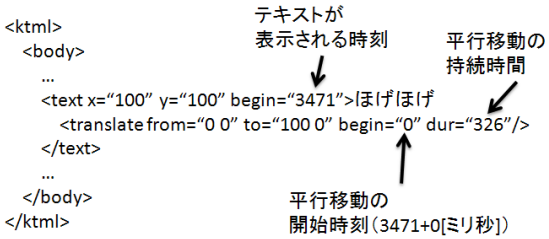

図 3 KTDL におけるアニメーションの開始時刻と持続時間の記述
Fig. 3 Description of beginning and duration time in KTDL.

特に，曲中の全ての音に対して文字アニメーションとタイミングを合わせる場合は作業量が増え，計測に多くの時間を費やさなければならない．

3. 文字アニメーション記述言語：おどりたい

2. で述べたように，既存の手法では音楽と文字アニメーションの同期をとる場合に音の開始時刻と持続時間を計測する手間がかかる．そこで本研究では，文字アニメーション記述言語における時間の記述において，拍に基づいた記述法を提案する．

3.1 拍 記 述

音楽にはリズムが存在し，そのリズムは一定時間ごとに刻まれる拍という単位によって作られる．KTDL のように音の長さを時間で表現するということは，拍という単位を時間の

単位に変換するということであり，そのために計測の手間が発生する．音楽に合わせた文字アニメーション映像を作るのであれば，音楽が本来持つ拍を単位として音の長さを記述することで，簡単に記述ができると考えられる．以降，拍に基づいた音の長さの記述のことを“拍記述”，KTDL のような時間に基づいた記述のことを“時間記述”と呼ぶ．

拍記述は拍を用いるという音楽に合わせた記述であるため，楽譜の記法を模倣した記述法にすることにより，曲の楽譜を書くイメージで文字アニメーション映像を作成できると考えられる．ここで，楽譜は音符や休符を時系列順に並べたものであり，それぞれが何拍継続するかということのみを示す．そして，並べられた順に逐次的に音を演奏することを意味する．このとき，特定の音が鳴るのは曲が始まってから何拍後か，という表現はしない．

そこで，拍記述は時間記述と違い開始時刻を考えない．その代わり，全ての文字アニメーションを時系列順に並べ，各持続時間を拍数で記述することで文字アニメーション映像を表現する．したがって，拍記述では開始時刻と持続時間を計測する手間を省くことができる．

以上の指針に基づき，時間を拍数で表現する文字アニメーション言語,「おどりたい」を設計した.「おどりたい」は KTDL を基に，時間の記述方法を前述した拍記述に変更したものである．

3.2 「おどりたい」の設計

この節では「おどりたい」で記述したソースコード（**図 4**）を例に,「おどりたい」の構造，主要な要素について述べる．

(1) おどりたい文書の全体構造

図 4（1）で示した部分がおどりたい文書の全体構造である．`<dtml>` 要素はこの文書がおどりたい文書であることを示し，1 つの文書に 1 つ必要となる．`<body>` 要素は自身の子要素がおどりたい文書の本文であることを示す．おどりたい文書の本文は次項で述べるアニメーションテキストと`<rest>` 要素のセットで構成される．また，`<body>` 要素の `tempo` 属性では曲のテンポを bpm（beat per minute）単位で指定する．

(2) おどりたい文書における文字アニメーションの記述

`<text>` 要素内ではテキストとそのテキストに付与するアニメーションを記述する．1 つのテキストに対してアニメーションは 1 種類または複数種類付与することができる．`<text>` 要素にテキストとアニメーションを記述したものを以降，“アニメーションテキスト”と呼ぶ．図 4（2）で`<text>` 要素内に記述されているのは“毎日”というテキストと`<translate>` 要素であり，これはテキストを平行移動させる要素

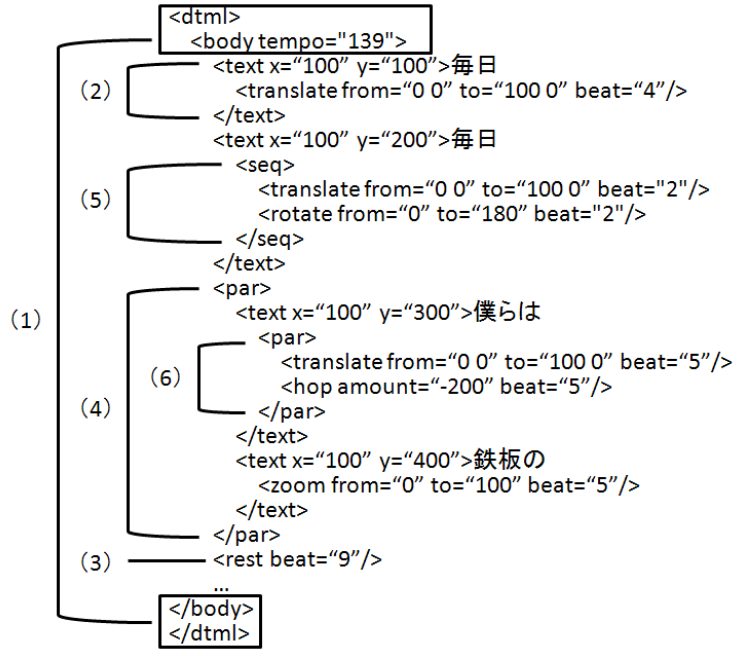

図 4 「おどりたい」ソースコードの例
Fig. 4 An example code of Odoritai.

である．

`<translate>` のような，動きを表す要素のことを“単位アニメーション”と呼ぶ．単位アニメーションは予め用意されているアニメーションであり，平行移動，ホップ，フォントサイズを使った拡大縮小，x 方向，y 方向の倍率を使った拡大縮小，回転，色の変化と透明度の変化の計 7 つがある．

「おどりたい」の`<translate>` 要素には，図 3 と違い，`begin` 属性と `dur` 属性の代わりに `beat` 属性が記述されている．この `beat` 属性に拍数を記述することで，テキストがその拍数の間動きを継続する．`beat` 属性に記述された拍数は`<body>` 要素の `tempo` 属性で指定されたテンポにおける拍の長さとなる．すなわち，図 4（2）は“毎日”というテキストが平行移動を 4 拍継続する文字アニメーションを表現している．

(3) **おどりたい文書における空白時間の記述**

「おどりたい」では，文字が表示されない時間や文字が動かない時間を“空白時間”と呼ぶ．この空白時間を表すには<rest> 要素を用いる．これは楽譜の記法での休符に相当する．<rest> 要素中の beat 属性に拍数を記述することで，その拍数分何も表示されない，または何も動かない時間を表現することができる．図 4 (3) では，空白時間が 9 拍あることを表している．

(4) **アニメーションテキストの逐次実行と並列実行**

楽譜では 1 つの五線譜に並べられた音符は全て逐次的に鳴らされることを意味する．「おどりたい」においても，並べられたアニメーションテキストは全て逐次処理される．しかし，一般的な文字アニメーションには複数の文字を同時に動かす表現が現れる．楽譜上で複数の音を同時に鳴らすことを表現するには，新たな五線譜を追加するか，複数の音符を垂直に並べて和音とする．前者は，五線譜を表すまとまりごとにアニメーションテキストを書き分けるという表現になる．しかし，この構造だとまとまり同士の間でどの文字アニメーションが同時に実行されるかが分かりにくい．そこで,「おどりたい」では後者を模した表現方法を用いる．

和音は，特定の時点で複数の音を一斉に鳴らすことを意味している．その特定の時点を表現するために，<par> 要素を用いる．すなわち，各アニメーションテキストを同時に表示したり動かしたりする場合は，それらを<par> 要素内に記述する．例えば図 4 (4) では，“僕らは”というテキストと“鉄板の”というテキストが同時に表示され，同時に動かされることを表現している．

一方，逐次実行を明示的に記述したい場合は，アニメーションテキストを<seq> 要素内に記述する．<body> 要素の直下に複数のアニメーションテキストを記述した場合は，暗黙的に<seq> 要素内に記述されたことになる．

(5) **単位アニメーションの逐次実行と並列実行**

KTDL では 1 つのテキストに対して複数の単位アニメーションを割り当てる場合，単位アニメーションの開始時刻を変えることで，テキストが各アニメーションを逐次，または並列に実行することを表現する.「おどりたい」では前述のとおり，アニメーションテキストの逐次実行，並列実行に<seq> 要素，<par> 要素を用いる．この要素を単位アニメーションの逐次実行と並列実行にも用いる．すなわち，1 つのテキストが逐次的に異なる動きを行うことを記述する場合は，単位アニメーションを<seq> 要素内に記述する．図 4 (5) では，“毎日”というテキストが平行移動を 2 拍継続した後，回転を 2 拍継続するという文字アニメーションを表現している．

1 つのテキストが複数の動きを同時に行うことを記述する場合は，単位アニメーションを<par> 要素内に記述する．図 4 (6) では，“僕らは”というテキストが平行移動しながらホップするという文字アニメーションを表現している．

なお，<text> 要素の直下に複数の単位アニメーションを記述した場合は，暗黙的に<seq> 要素内に記述されたことになる．

3.3 文字アニメーション映像生成ソフトウェアの実装

3.2 で述べたおどりたい文書から文字アニメーション映像を生成するために，おどりたい文書を文字アニメーション映像に変換するソフトウェア「おどりたいエディタ」を実装した．おどりたい文書を文字アニメーション映像に変換するために Lee ら[5] の Kinetic Typography Engine（KTE）を採用した．KTE はテキストを動かすためのクラスのライブラリと再生エンジンからなっており，KTML を文字アニメーション映像に変換するソフトウェアに採用されている．

実装したソフトウェアは，おどりたい文書を XML 文書として解析し，要素ごとに KTE で読み込める形に変換して，再生エンジンに渡す．そして再生エンジンを用いて文字アニメーション映像を再生する．

4. 「おどりたい」と KTDL の比較評価

本章では,「おどりたい」の有用性を評価する実験について述べる．

4.1 実験タスク

本実験では被験者に,「おどりたい」と KTDL それぞれで 1 つずつ文字アニメーション映像を作成させる．被験者に作成させる映像のお手本を“易しい”と“難しい”の 2 種類を作成した．難易度の設定には拍数の細かさに着目し，“易しい”のお手本は，単位アニメーションの拍数をすべて 1 とした．一方，“難しい”のお手本は，単位アニメーションの拍数の最小単位を 0.25 とし，各単位アニメーションの拍数を異なるものにした．この 2 つの条件について KTDL との比較評価を行う．お手本の詳細を表 1 に示す.

4.2 手　　順

被験者は情報工学を専攻する大学生，大学院生 12 人で，この 12 人を 6 人ずつの 2 グループ（α，β）に分けた．また，全員 HTML を手書きした経験があった．グループ α の被験者には“易しい”のお手本を見て，同じものを「おどりたい」と KTDL でそれぞれ 1 回ずつ作るように指示し，グループ β の被験者には“難しい”のお手本を見て，同じものを「お

どりたい」と KTDL でそれぞれ 1 回ずつ作るように指示した．使用する言語の順番はカウンタバランスを考慮した．各試行の手順は以下のとおりである．

まず,「おどりたい」で作成させる場合を述べる．最初に，被験者に「おどりたいエディタ」の使用法に関するドキュメントと「おどりたい」の言語仕様を全て読ませた．次にお手本の映像を 1 回見せてから，曲のテンポとタスク終了のための基準を紙面で与え，映像作成を開始させた．本実験では，被験者がタスク終了のための基準を全て満たしていると判断した時点でその旨を実験者に報告し，試行を終了した．被験者に与えた基準は以下のとおりである．

(a) 映像中で使用されている単位アニメーションの種類が全て一致していること
(b) 映像中で使用されているテキストが全て一致していること
(c) 作成した映像の音楽と文字アニメーションのタイミングをお手本の通り一致させること
(d) 表示される各アニメーションテキストの座標とフォントサイズのお手本との差がそれぞれ± 50pixel，± 20% に収まっていること（なお，作成する文字アニメーション映像の描画領域は 640 × 480pixel である）

なお，映像作成を開始してから，被験者は自由にお手本の映像を閲覧することができた．また，被験者には電卓を与え，使用を許可した．

次に，KTDL で作成させる場合を述べる．手順やタスク終了のための基準は「おどりたい」の場合と同じで，使用する言語を KTDL に変えた．また被験者にテンポは与えず，タスクを開始させる前に実験者が波形表示ソフトウェアの説明を行い，再生，一時停止，巻き戻し，頭出し，開始位置変更の操作を全て被験者に練習させた．そして，音楽と文字アニメーションのタイミングを合わせる際には，波形表示ソフトウェアに表示される時間を見て KTDL の開始時刻と持続時間を記述するように指示した．

4.3 結　　果

本実験では所要時間と，お手本との差を評価尺度として用いる．所要時間とは被験者に映像作成を開始させた時点から，自身の作品が基準を全て満たしていると被験者が判断した時点までのことをいう．お手本との差はお手本との時間的なずれの観点から考える．このため，お手本のソースコードと被験者が作成したソースコードを比較し，時間記述ならアニメーションテキストの開始時刻と持続時間，拍記述ならアニメーションテキストの持続時間と休符の拍数の差を見る．時間や拍を記述する箇所のうちお手本と差がある箇所の割合をお手本との品質の差の指標とし，以降“お手本との差の割合”と呼ぶ．お手本との差の割合が高いほど，お手本より品質が劣ることになる．両お手本合わせてアニメーションテキストに用いられる最小の刻みが 0.25 拍なので，お手本との差が 0.25 拍以上（時間記述なら 0.25 拍を時間に直した値以上）ある箇所を数える．平均所要時間を**図 5** に，お手本とのずれの割合の平均を**図 6** に示す．

表 1　実験で用いたお手本の詳細
Table 1　Details of examples given in the experiment.

難易度	単位アニメーションの拍数	使用した曲	曲の範囲
易しい	整数かつ全て同じ	たいせつなもの	0 分 00 秒〜0 分 18 秒
難しい	実数かつ異なる	勝手にシンドバッド	0 分 00 秒〜0 分 27 秒

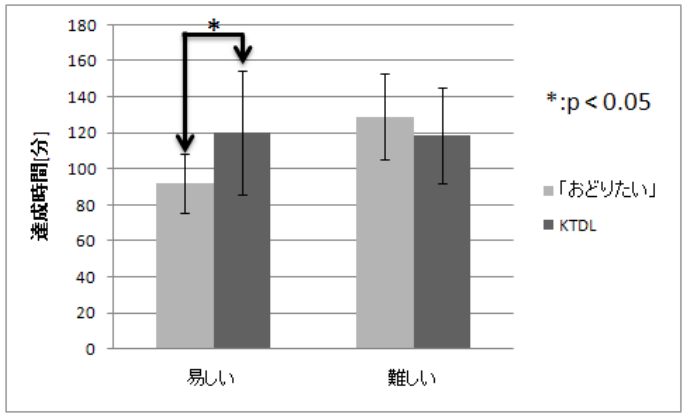

図 5　平均所要時間
Fig. 5　Completion time

図 5 より，“易しい”の映像作成では,「おどりたい」のほうが KTDL よりも有意に平均所要時間が短かった．これに対し，“難しい”の映像作成については「おどりたい」のほうが平均所要時間が長かったが，有意差は見られなかった．

図 6 より，“易しい”の映像作成では「おどりたい」を用いると，KTDL より正確に作れることが分かった．また，“難しい”の映像作成については両手法間に有意な差が見られなかった．

4.4 考　　察

図 5，図 6 より，グループ α では「おどりたい」のほうが KTDL より平均所要時間が有意に短く，お手本との差の割合が有意に低い．“易しい”の映像を「おどりたい」で記述す

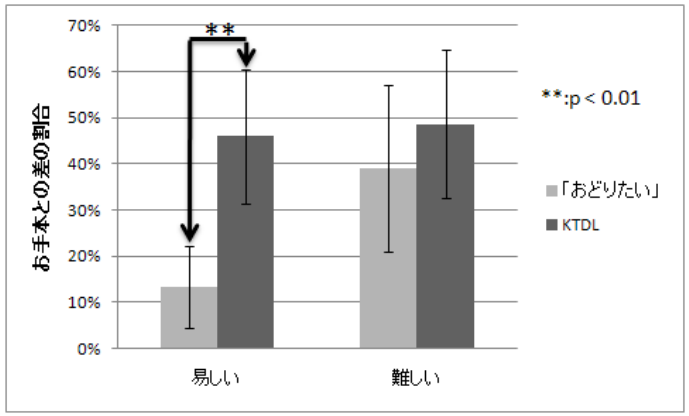

図 6 お手本との差の割合
Fig. 6 Rate of difference from example

る場合，各単位アニメーションの拍数と曲から感じられる拍（4 分音符の長さ）とが一致していたため，拍数を容易に認知でき，素早く記述できたと考えられる．また，グループ α の被験者の 1 人は「おどりたい」を用いて基準を完全に満たした作品を作ることができた．それに対して KTDL では，全てのアニメーションが同じ時間とはいえ，開始時刻を波形表示ソフトウェアで 1 つ 1 つ計測する必要があり，この計測に時間を要したと考えられる．

また，KTDL で音楽と完全にタイミングを合わせた文字アニメーション映像を作ることは難しいことがわかった．これを示す結果として，グループ α ではどの被験者も KTDL のほうが「おどりたい」よりお手本との差の割合が高かったことが挙げられる．これは被験者のうち，“ 易しい ” の映像を KTDL で作る場合に全てのアニメーションが同じ時間であることを利用して最初のアニメーションだけ時間を測り，その時間を残り全てに適用する方法を採った者がいたことが原因のひとつと考えられる．この方法を用いると持続時間を 1 つ 1 つ計測する手間は省けるが，最初に測った持続時間にお手本の持続時間と差があると，その値を利用する単位アニメーションが全てお手本とずれる．この方法を採った被験者の多くは，計測した持続時間の値がお手本の持続時間と 0.25 拍以上の差があったことに気づかずに多くの単位アニメーションにその値を適用したため，「おどりたい」に比べて KTDL のほうがお手本との差の割合が高くなった．

グループ β では「おどりたい」のほうが KTDL より平均所要時間が長かったが有意差は見られなかった．また，図 6 より，グループ β の「おどりたい」におけるお手本との差の割合は，グループ α のものより高い．“ 難しい ” の映像を「おどりたい」で記述する場合，各単位アニメーションの拍数と曲から感じられる拍（4 分音符の長さ）とが異なるため，拍数を認知しづらくお手本と一致させるのに時間を要したと考えられる．一方 KTDL では，所要時間とお手本との差の割合がともに “ 易しい ” の映像における KTDL の所要時間とお手本との差の割合と同程度である．KTDL では難易度によらず，時間を計測する手間は同じである．すなわち，時間を直接表現する KTDL において拍の細かさは所要時間と質に影響しないことが言える．

5. おわりに

本研究では，文字アニメーション記述言語で音楽に合わせた文字アニメーション映像を簡単に作ることを目指した．そのために，拍記述に基づいた文字アニメーション記述言語「おどりたい」を設計し，「おどりたい」を文字アニメーション映像へ変換するソフトウェアを開発した．また，拍数の細かさと拍数が一定かどうかという 2 つの要因を用いて難易度を決め，KTDL との比較実験を行った．

その結果，「おどりたい」は単位アニメーションの拍数が整数かつ一定である文字アニメーション映像を，KTDL よりも早く正確に作れることが分かった．

今後は，曲の条件を変えて「おどりたい」の有用性を評価する．例えば，途中からテンポが変わる曲や 3 連符や 5 連符などの連符を多用する曲など，実際の曲で起こる条件の基で調査する必要がある．

謝辞 本研究の一部は科研費 (20500120) の助成を受けたものである．

参 考 文 献

1) Z. Yeo, Kinetic Typography: Generation and Application of Affective, Animated Text,（オンライン）, 入手先（http://src.acm.org/2010/ZhiquanYeo/Yeo/Yeo.html）（参照 2011-2-7）.
2) 水口 充，田中 克己：文字アニメーションの自動合成の試み，情報処理研究報告，vol.2005-HI-116，pp.97–104（2005）.
3) ニコニコ動画（オンライン），入手先（http://www.nicovideo.jp/）（参照 2011-2-7).
4) YouTube（オンライン），入手先（http://www.youtube.com/）（参照 2011-2-7).
5) J.C.Lee, J.Forlizzi, and S.E.Hudson: The kinetic typography engine: An extensible system for animating expressive text, UIST 2002, pp.81–90 (2002).