VSML/VSS: テキストベース動画編集のための動画構造/装飾記述言語

志岐 颯駿[1,a)]　江本 健斗[1]

概要：動画配信プラットフォームの発達により，個人が動画を制作し投稿することが多くなった．特に，個人の興味に特化したコンテンツを字幕付きで解説するような動画が多い．しかし，そのような動画は解説のための大量の文字列・音声等のオブジェクトを含むため，例えば字幕表示の見栄え等を推敲する際には多量のオブジェクト設置や装飾変更の操作が必要となり，既存の動画編集アプリケーション等での GUI による編集では手間が大きすぎるという問題がある．本研究は，この問題点を解決するため，動画の「構造情報」と「装飾情報」とを分離し，オブジェクトの時間的・空間的配置を相対的な関係として記述する，動画構成記述言語 VSML/VSS を提案する．

キーワード：XML, 動画編集, 言語設計

VSML/VSS: Video Structure/Style Description Languages for Text-based Editing

Shiki Hayato[1,a)]　Emoto Kento[1]

Abstract: Recent development of video streaming platforms has resulted in many individuals creating and publishing their videos. In particular, there are many videos with subtitles that explain content specific to individual interests. However, editing such a video on ordinary GUI video-editing software suffers from tedious object manipulation. This is beause such a video contains a number of string/audio/image objects for kindful explanations and, for example, polishing subtitles requires precise move of many string/audio objects on the timeline and opening their properties to change their styles. In this research, to solve the problem, we proposes video structure/style description languages VSML/VSS that separate “structural information” and “decoration information” of a video and describe spacial and temporal arrangement of content objects as their relative relations.

Keywords: XML, video editing, language design

1. はじめに

近年，動画配信プラットフォームの発達により，個人が動画を編集して公開することが多くなってきた．例えば，最も広く知られる動画配信プラットフォームのひとつである YouTube では，毎分 500 時間分もの動画がアップロードされている [7].

[1] 九州工業大学
Kyushu Institute of Technology, Iizuka, Fukuoka 820–8502, Japan
[a)] h_shiki@pl.ai.kyutech.ac.jp

個人により公開される動画の様相は様々だが，多数の動画が制作・公開されるジャンルの一つとして “解説動画” がある．これは，動画制作者が個人的に興味を持ったコンテンツに対してその解説を音声・字幕付きで行う動画であり，動画配信プラットフォームのユーザ増加に伴い非常に広範囲のコンテンツに対する解説動画が公開され続けている．また，公開される動画は，画面構成や字幕のスタイル等の推敲を重ね，視聴に優しい動画になってることが多い．

解説動画のような多くの字幕や画面構成部品を含む動画の編集は，しかしながら，既存の動画編集アプリケーショ

ン等での GUI による編集では手間が大きすぎるという問題がある．これらの動画は必然的に多量の音声・文字列等のオブジェクトをその構成要素とするため，その編集作業においてはこれらの多量のオブジェクトを操作する必要がある．例えば，字幕の見栄えや切替タイミングを推敲する際には，タイムライン上で大量の字幕文字列オブジェクトのプロパティを開いて文字色等の属性を変更したり，大量の解説音声とそれに対応する字幕文字列オブジェクトに対して音声の再生終了から一拍後に字幕の表示を終了するように配置しなおしたり，それらの対を一定間隔を開けてタイムライン上に配置し直したりと，かなりの手間に悩まされる．

この大きな手間の主な原因は，以下の２点であると考えられる：

- 画像や音声などの各オブジェクトが関連を持たず独立してタイムラインに配置されていること．すなわち，それぞれのオブジェクトが，自身の絶対的な開始位置・終了位置の情報でタイムライン上に配置されるという点である．例えば，解説音声とそれに対応する字幕を表示する場合，その字幕の表示時間は音声データの長さに依存した長さにしたい．そのため，音声データを差し替える際には，同時に字幕の表示時間もその長さに応じて別途修正を加える必要が生じ，これは手間である．
- オブジェクトのサイズや色などの装飾情報が各オブジェクトごとに独立に保持され，包括的に装飾を指定することが出来ないこと．例えば，複数の字幕に対し，それらの色・サイズ・位置を同一に保ちつつ，これらの値を変えて見た目を推敲したいことがある．このとき，個々のオブジェクトが独立にこれらの情報を保持していると，それらを１つ１つ修正する必要が生じ，これは手間である．複数選択による装飾の同時変更も考えられるが，多少の手間を減らせるものの根本的な解決とはいえない．

これらの点を解消した動画編集手法が望まれる．

本研究は，上記の問題点の解決を目標に，動画構成記述言語 VSML/VSS を提案する．VSML/VSS は，動画内のオブジェクトの配置を階層構造として記述する「構造情報」と，それらの見た目を記述する「装飾情報」とを分離する．また，オブジェクトの時間的・空間的な配置をオブジェクト間の相対的な関係として記述する．VSML と VSS の関係は，ハイパーテキスト文書の記述に用いられる HTML と CSS の関係に相当し，一方を記述するときにもう一方を意識する必要がないというメリットがある．また，VSML/VSS から一般的な動画ファイルを生成する変換器を実装し，VSML/VSS による動画編集を複数名に実際に行ってもらい提案手法の評価を行った．結果，VSML/VSS により上記の問題点を大きく解消できることが確認された．

```
1 <elem>content1</elem>
2 <p><c1/><c2/></p>
3 <e attr="str_val">content2</e>
```

図 1 XML 文書の例

以降の本論文の構成は以下のとおりである．まず，第 2 節にて既存の事柄についての導入を行う．次に，第 3 節にて提案手法を導入する．そして，第 4 節にて提案手法の評価を示す．最後に，第 5 節にて関連研究を述べ，第 6 節にて本論文をまとめる．

2. 準備

本節では，本論文で用いる既存結果について導入する．

2.1 XML と XML スキーマ

XML（Extensible Markup Language） [4] は，W3C によって勧告された，汎用的な構文を定義した拡張が容易なマークアップ言語である．

図 1 に XML で記述された文書（XML 文書）の例を示す．XML 文書は，１つないし複数の木構造データを記述する．木構造データの各ノードは，要素と呼ばれ，その名前を *name* とすると，角括弧（< と >）でその名前を挟んだ開始タグ <*name*> から，同様に / を伴う角括弧で名前を挟んだ終了タグ </*name*> までの文字列で記述される．開始タグと終了タグの間には，そのノードの子が記述される．子には，同様の書式で記述される要素か，純粋な文字列が含まれる．また，子要素を持たない要素は，<*name*/> のように，閉じ角括弧の手前に / を置いた１つのタグで記述できる．例えば，図 1 の 1 行目は「“content1” という文字列を子に持つ要素 `elem` 」を記述し，2 行目は「子要素を持たない要素 `c1` と `c2` を子にもつ要素 `p`」を記述する．

要素は，子の他に，属性を持つことができる．属性は，名前付きの文字列であり，要素の開始タグの中に記述される．具体的には，その名前が *aname* で値が *val* であるならば，*aname*="*val*" が開始タグ中の要素名の後ろに記述される．例えば，図 1 の 3 行目は「“content2” という文字列を子に持ち，“str_val” という文字列を値とする属性 `attr` を持つ，要素 `e` 」を記述している．

XML を用いて独自の言語を定義する場合，与えられた XML 文書がその言語の文法（許容する要素や属性の構造）に合致しているかどうかを検証したい．そのための技術として XML スキーマ（XSD） [5] が利用される．様々なプログラミング言語上で，与えられた XML 文書が XSD で記述された構造を持つかどうかを検証する，XSD 検証器を利用できる．

表 1　HTML の代表的タグ

タグ名	用途
h1	1 番大きい見出し
p	パラグラフの表現
div	純粋なコンテナ（ブロックボックス）
span	純粋なコンテナ（インラインボックス）
img	画像の表示
br	文章の改行

2.2 HTML/CSS

HTML（HyperText Markup Language） [6] は，ハイパーテキスト文書（ウェブページ）を表現するためのマークアップ言語である．HTML はもともとハイパーテキスト文書内のコンテンツの構造と装飾の両方を記述するものとして利用されていたが，後述する CSS の利用が広まるにつれ，HTML はハイパーテキスト文書内のコンテンツの構造，すなわち，「何のコンテンツの内側に何のコンテンツがあるか」や「何のコンテンツと何のコンテンツが並んでいるか」といった，コンテンツ間の相対的な関係のみを記述するようになった．

HTML の書式はタグを用いて要素を記述する XML と同様であり（正確には XML の親に当たる SGML をベースとする），ハイパーテキスト文書を記述するために，画像や他のハイパーテキスト文書へのリンクなどを記述するタグを持つ．表 1 に代表的な HTML タグを示す．HTML の要素は決まった要素名を持つため，以降，決まった要素名をタグ名と呼ぶ．

HTML には，そのコンテンツ表示のためのボックスモデルという概念があり，HTML のコンテンツにはブロックボックスとインラインボックスの 2 種類がある．ブロックボックスは，横幅が親要素の幅で表示される．インラインボックスは，自要素の幅で表示される．HTML のコンテンツは，基本的にそれが記述されるタグによって，ブロックボックスかインラインボックスかが決まっている．

CSS（Cascading Style Sheet） [1] は，HTML で記述されたハイパーテキスト文書（以下，HTML 文書）の装飾を記述するためのスタイルシート言語である．

CSS は，タグ名・クラス（class 属性で指定した値）・ID（id 属性で指定した値）およびそのネスト構造に対して，それに適用する装飾を記述する．装飾は，プロパティ名とその値となる文字列とのペアの集まりとして記述される．例えば，文字サイズは font-size という名前のプロパティで記述され，背景色は background-color というプロパティで記述される．

図 2 に CSS の具体例を示す．CSS は，装飾の対象を表すセレクタの後ろに，その対象に適用する装飾の具体値を { と } で囲んで記述する．セレクタは，装飾の対象がタグ名であればそのタグ名を，対象がクラスであれば ‘.’ に続けてクラス名を書いて指定する．装飾の具体値は，指定したいプロパティとその値のペアを ; で区切って列挙して記述する．各ペアは，プロパティ名 *pname* に対して値 *val* を指定したい場合，*pname*:*val* と記述される．以上を踏まえ，図 2 の CSS は，`body` タグのコンテンツに対する「背景色を白，文字サイズを 12pt とする」という装飾と，large クラスのコンテンツ（すなわち，タグに指定された `class` 属性の値が “large” である要素）に対する「文字サイズを 24pt とする」という装飾を定義する．

```
1 body {
2   background-color: white;
3   font-size: 12pt;
4 }
5 .large {
6   font-size: 24pt;
7 }
```

図 2　CSS の例

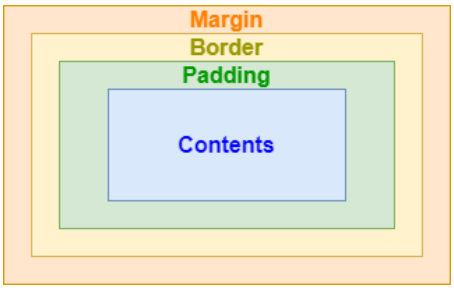

図 3　CSS ボックスモデル

CSS には CSS ボックスモデルという概念があり，ボックスは，コンテンツボックス・パディングボックス・ボーダーボックス・マージンボックスの 4 要素から構成される．CSS ボックスモデルを図 3 に示す．

コンテンツボックスはコンテンツが表示される範囲であり，パディングボックスはコンテンツボックスの周囲にある余白である．ボーダーボックスはコンテンツボックスとパディングボックスを囲う，ボーダーライン描画の範囲であり，マージンボックスはこのボックスと他のボックスとの間の余白である．

CSS のプロパティには，コンテンツボックスの幅・高さを指定する width・height，パディングの幅を指定する padding，ボーダーの幅を指定する border，マージンの幅を指定する margin が含まれる．他，主要なプロパティを表 2 に示す．

3. VSML/VSS の提案

本節では，本研究の提案する動画構成記述言語 VSML/VSS を導入する．さらに，VSML/VSS から一般的な動画ファイルを生成する変換器について述べる．

表 2　CSS の代表的プロパティ

プロパティ名	用途
display	要素の表示指定
width	コンテンツの幅指定
height	コンテンツの高さ指定
padding	パディングの幅指定
border	ボーダーの幅指定
margin	マージンの幅指定
color	コンテンツの色指定
background-color	コンテンツ，パディングの背景色の指定
font-size	文章の文字サイズ
font-family	字幕のフォント指定
font-weight	ボールドの指定
font-style	イタリックの指定

3.1　動画構造記述言語 VSML

動画構造記述言語 VSML（Video Structure Markup Language）は，動画内のオブジェクトの構造，すなわち，時間的・空間的にオブジェクトがどのように配置されるのかを，オブジェクトの入れ子関係や並びなどの関係によって記述する XML ベースの言語である．

表 3 に VSML のタグ一覧を示す．VSML のタグは，大きく 2 種類に分かれ，静止画・動画・音声・文字列の基本オブジェクトを表すタグと，それらの時間的・空間的な配置を表すタグとがある．また，XML ベースであるため，オブジェクトの入れ子構造はタグの入れ子構造として自然に表現される．

基本オブジェクトを表すタグとして，文字列を表示する `txt`，静止画を表示する `img`，音声を再生する `aud`，動画を再生する `vid` がある．これらは，それぞれ時間軸方向の基本長さを持つ．具体的には，文字列と静止画は無限の長さを，音声と動画はそれぞれの再生時間の長さを持つ．また，静止画・動画・文字列は空間的な大きさを持つ．

オブジェクトを時間的・空間的に配置した複合オブジェクトを構成するためのタグとして，`seq`，`rect`，`prl`，`layer` がある．表 4 にこれらのタグが子要素を空間的・時間的にどの様に並べるかを整理して示す．オブジェクトの並びの関係には，時間軸方向と空間軸方向とに，それぞれ順に並べるのか同じ位置に重ねるのかの選択がある．

`seq` と `rect` は，子要素を時間的に並べ空間的に重ねる．すなわち，すべての子要素が同じ位置に，それぞれの長さずつ順番に表示・再生される．例えば，音声を順番に再生したり（図 4），シーンを並べて次々と切り替えて再生するのに使われる．`seq` と `rect` の違いは，`seq` は背景を持たず，`rect` は背景を持つことである．これらの要素の空間的なサイズは，子要素の幅・高さの最大値となる．

`prl` は，時間的にも空間的にも子要素を重ねる．`prl` 要素の時間的長さは，有限長の子要素の時間長さの最大値となる．例えば，音声の再生と字幕の表示を同時に行うには，図 5 のように記述する．これにより，「音声の再生とともに，その間ずっと字幕が表示される」という複合オブジェクトが出来上がり，その時間的長さは音声の再生時間長となる．また，子要素は空間的に重ねられるので，図 6 のように記述することで背景の上に字幕を表示できる．

`layer` は，子要素を時間的に重ね空間的に並べる．`layer` 要素の時間的長さは，`prl` と同じく，有限長の子要素の時間長さの最大値となる．例えば，画像を横並びに表示したい時は，図 7 のように記述する．子要素の並びの方向は，後述する VSS による装飾として指定できる．

表 3　VSML のタグ

タグ名	用途	標準の再生時間
txt	字幕の表示	無限
img	画像の表示	無限
aud	音声の再生	音声データの長さ
vid	映像の再生	映像データの長さ
seq	子要素を順番に再生	子要素の再生時間の総和
rect	子要素を順番に再生	子要素の再生時間の総和
prl	子要素を重ねて同時に再生	子要素の再生時間の最大値
layer	子要素を並べて同時に再生	子要素の再生時間の最大値

表 4　タグの時間的・空間的配置

		空間的	
		重ねる	順に並べる
時間的	重ねる	prl	layer
	順に並べる	seq, rect	

```
1 <seq>
2   <aud src="こんにちは.mp3" />
3   <aud src="私は元気です.mp3" />
4 </seq>
```

図 4　seq タグによるオブジェクトの時間的配置の例

```
1 <prl>
2   <txt>こんにちは</txt>
3   <aud src="こんにちは.mp3" />
4 </prl>
```

図 5　prl タグによる字幕と音声の同時再生の例

```
1 <prl>
2   <img src="flower_garden.png" />
3   <txt>これは花畑の画像です</txt>
4 </prl>
```

図 6　prl タグによる空間的重ねの例

```
1 <layer>
2   <img src="cat.png" />
3   <img src="dog.png" />
4 </layer>
```

図 7　layer タグによる空間的配置の例

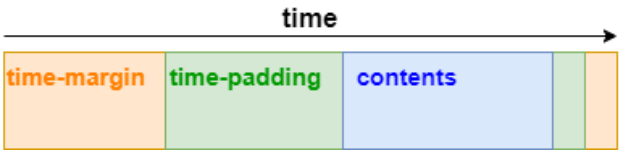

図 8　VSS 時間的ボックスモデル

3.2　動画装飾記述言語 VSS

動画構造記述言語 VSS（Video Styling Sheet）は，VSML で記述された動画オブジェクトの装飾を記述する，CSS をベースとした言語である．CSS と同様に，セレクタにより装飾対象を指定し，その対象に適用する装飾をプロパティとその値のペアの集まりとして記述する．

表 5 に VSS のプロパティを示す．

画面上の描画の装飾は，CSS ボックスモデルをそのまま導入し，margin，border，padding，width，height のプロパティで CSS 同様に装飾を指定する．また，動画には時間軸の概念が存在するため，CSS ボックスモデルの概念を時間軸方向にも展開し，図 8 のようにオブジェクトの時間的前後の余白を指定するプロパティ time-margin と time-padding をもつ．padding と同様に，time-padding の範囲の時間的余白では背景色指定などは有効となる．

フォントや色に関するプロパティは CSS を踏襲する．動画に関する独自のプロパティとしては，オブジェクトの再生時間を指定する object-length や，子要素の時間的・空間的重なりを制御する order・layer-mode をもつ．さらに，音声に関するプロパティとして，音量を指定する audio-volume や，モノラル／ステレオを指定する audio-system などを持つ．

なお，order や layer-mode を用いると，`seq`・`prl`・`layer` の振る舞いを互いに互いのものに変更することができる．これらのタグは，デフォルトとしてもつ order・layer-mode プロパティの値が異なるという関係にある．同様に，`seq` と `rect` は，デフォルトの background-color プロパティ値が違うだけという関係にある．この辺りの状況は HTML/CSS の状況と同じである．

3.3　VSML/VSS からの一般的動画形式への変換

本研究では，前節までに示した VSML/VSS での記述から一般的な形式で動画を出力する，変換器の実装を行った．変換器は Python で実装されており，動画エンコードのバックエンドには FFmpeg を用いた．

変換器のフローを図 9 に示す．まず，XML スキーマにより入力 VSML ファイルを検証し，検証をパスした DOM オブジェクトを得る．次に，入力 VSS ファイルを正規表現で検証し，DOM オブジェクトをたどりつつ，VSML の木構造を表現する内部オブジェクトを構築する．最後に，内部オブジェクトの木構造を再帰的にたどりつつ，FFmpeg に渡すコマンドを組み立て，FFmpeg に動画を出力させる．FFmpeg に渡すコマンドの構築は素直にできる．

表 5　VSS のプロパティ

プロパティ名	用途
width	オブジェクトの幅の指定
height	オブジェクトの高さの指定
margin	マージンの幅指定
padding	パディングの幅指定
background-color	背景色の指定
font-family	字幕のフォント指定
font-size	字幕の文字サイズ指定
font-weight	ボールドの指定
font-style	イタリックの指定
object-length	オブジェクトの時間的長さの指定
time-margin	前後の時間的余白の指定
time-margin-start	前の時間的余白の指定
time-margin-end	後ろの時間的余白の指定
time-padding	前後の時間的余白の指定
time-padding-start	前の時間的余白の指定
time-padding-end	後ろの時間的余白の指定
order	子要素の並べ方の指定
layer-mode	子要素を重ねるか並べるかの指定
direction	並べる際の方向指定
audio-volume	音声の音量指定
audio-system	モノラル/ステレオの指定
font-color	字幕の色指定
font-border-color	字幕の縁取りの色指定
font-border-width	字幕の縁取りの太さ指定
source-loop	ソースをループするかの指定
playback-speed	再生速度の指定
opacity	透明度
chroma-key	クロマキーの色指定
magnification	拡大率
rotate	回転量
border	ボーダーの幅指定
border-radius	境界の角丸めの半径指定

図 9　VSML 変換器の構造

実装した変換器のソースコードを Github に公開している：`https://github.com/PigeonsHouse/VSML`．

4.　評価実験

本節では，前節で設計・実装した VSML/VSS に関する評価を示す．

4.1　実験方法

被験者に対して，用意した 2 種類の動画編集作業を行ってもらい，アンケートを実施して VSML/VSS とその変換器の評価を行った．動画編集作業は大きく 2 つあり，従来手法である動画編集アプリケーションと VSML で同内容

表 6　実験に用いた計算機環境

OS	Windows 11 Pro 21H2 22000.2538
CPU	11th Gen Intel(R) Core(TM) i7-1185G7 @ 3.00GHz
RAM	32GB (4GB x8, LPDDR 4)
SSD	SAMSUNG MZVL22T0HBLB-00B00, NVMe, 2TB

の動画修正を行う作業と，VSML/VSS での動画編集作業を用意した．

従来手法の動画編集アプリケーションとして，Aviutl [8] を使用した．Aviutl のバージョンは version 1.10 で，プラグインには，拡張編集プラグイン version 0.92，SplitWindow 4.6.0，x264guiEx 3.27，L-SMASH Works r940 release1 を使用した．アンケートでは，各被験者の動画編集の経験量，HTML/CSS の習得度合いの情報も収集した．

本実験での動画編集作業は，表 6 に示す性能のノート PC で行ってもらった．

4.2　実験１：修正作業

この実験では，提案手法と既存手法のそれぞれで，3 つの動画修正タスクに取り組んでもらった．動画編集を行う際には，操作ミスや推敲で一度制作した動画のデータを編集する作業が発生する．この実験は，提案手法により，その際の作業時間が削減できることを確認するためのものである．よって，既に制作してあるデータから修正を行う作業をタスクとして設定した．

4.2.1　取り組んでもらうタスクとアンケート項目

取り組んでもらう 3 つのタスクは以下のとおりである：

タスク 1　同じ再生時間となるように時間的に整列している文字列・背景・音声オブジェクトの組が一定間隔で配置されている “時間的に整列された” 動画に対し，その一つの組の音声オブジェクトを異なる長さのオブジェクトに差し替え，全体を再び “時間的に整列された” 状態になるようオブジェクトの時間軸上の配置を修正する．

タスク 2　新しい音声オブジェクトとそれに対応する文字列オブジェクトを，すべての音声と音声の間隔が一定になるように既存の音声オブジェクトの並びの途中に挿入する．

タスク 3　動画内に登場する 2 人のキャラクターの字幕のそれぞれの縁取りの色を，初期の色から異なる色に変更する．

作業後に回答してもらうアンケートの項目は，従来手法にあった「同じ内容を維持したい装飾を一括で変更する手間がある点」,「オブジェクト間にオブジェクトを挿入する手間がある点」,「オブジェクトの始点終点を揃える手間がある点」の 3 つの問題点に対して，その解消度合いを「全く解消されていなかった」,「ほとんど解消されていなかった」,「よくわからなかった」,「概ね解消されていた」,「ちゃんと解消されていた」の 5 段階で尋ねた．

表 7　タスク 1 の操作時間（秒）とその軽減量（%）

被験者番号	従来手法	提案手法	軽減量
1	802	158	80.3
2	358	447	-25.0
3	809	472	41.6
4	646	91.8	85.8
5	626	121	80.7
6	731	117	84.0
7	667	169	74.7
平均値	663	225	60.3

表 8　タスク 2 の操作時間（秒）とその軽減量（%）

被験者番号	従来手法	提案手法	軽減量
1	695	178	74.3
2	195	209	-7.01
3	411	126	69.3
4	195	42.4	78.2
5	488	177	63.7
6	205	154	25.0
7	510	376	26.1
平均値	385	180	47.1

4.2.2　結果と考察

7 人の被験者の，従来手法と提案手法での各タスクの実行に要した操作時間を表 7，表 8，表 9 に示す．

ほぼ全ての被験者において，タスクの実行に要する操作時間は提案手法により大きく減少した．提案手法による従来手法からの操作時間の減少量は，全タスクでの平均で 53%である．このことから，提案手法である VSML/VSS を使用することで動画編集での修正作業の時間を減らすことが出来ていると言える．

唯一，被験者 2 だけは，提案手法による操作時間が既存手法よりも長い．これは，被験者 2 が従来手法の Aviutl の操作に慣れており，その操作を素早く行える一方で，慣れない提案手法に対してはその操作に時間がかかったものと思われる．ただ，その操作時間の差は大きくないため，提案手法は既存手法よりも，効率的な操作に慣れを必要とぜず直感的であるとも考えられる．

アンケートの回答を図 10，図 11，図 12 に示す．回答には「従来の問題点がちゃんと解消されている」・「概ね解消されている」という意見が多く，提案手法は従来手法の問題点であった「同じ内容を維持したい装飾を一括で変更する手間がある点」,「オブジェクト間にオブジェクトを挿入する手間がある点」,「オブジェクトの始点終点を揃える手間がある点」を解消できる手法であると考えられる．

4.3　実験２：制作作業

この実験では，VSML/VSS を用いてゼロから新たな動

表 9　タスク 3 の操作時間（秒）とその軽減量（%）

被験者番号	従来手法	提案手法	軽減量
1	432	94.5	78.2
2	119	130	-9.24
3	269	121	55.2
4	278	91.8	67.0
5	294	208	29.3
6	223	91.9	58.8
7	534	55.3	89.6
平均値	307	113.1	52.7

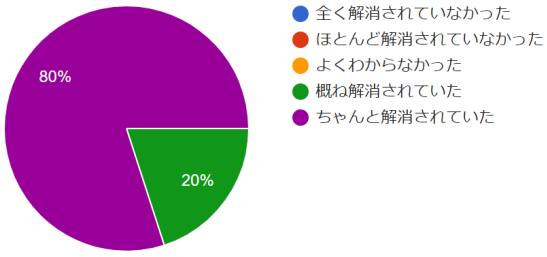

図 10　装飾を変更する手間

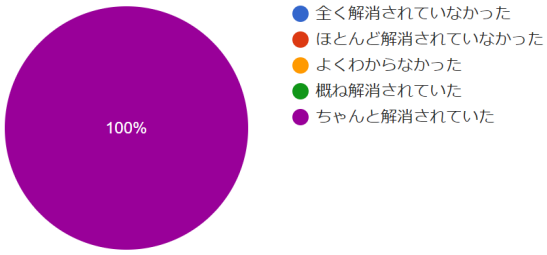

図 11　オブジェクト間にオブジェクトを挿入する手間

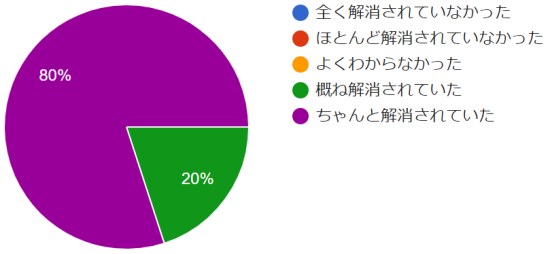

図 12　オブジェクトの始点終点を揃える手間

画を制作してもらう作業に取り組んでもらった．これは，VSML/VSS の習得難易度や使いやすさを測ることを目的とする．

4.3.1　取り組んでもらうタスクとアンケート項目

取り組んでもらう 3 つのタスクは以下のとおりである：

タスク 1　画像・音声・字幕を同時に再生する映像を作成する．

タスク 2　タスク 1 と同様に画像・音声・字幕を同時に再生するブロックを作成し，その 2 つのブロックを順番に再生する映像を作成する．

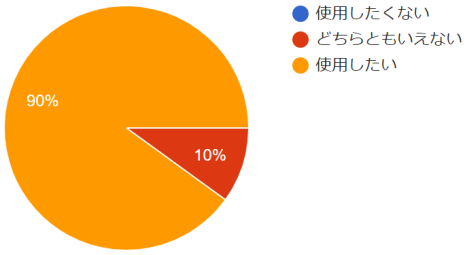

図 13　VSML を使用してみたいか

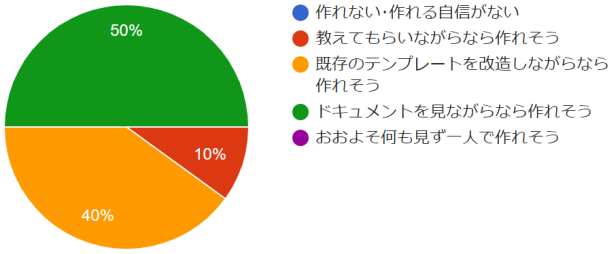

図 14　VSML で動画を制作できそうか

タスク 3　タスク 2 で作られた映像の裏で並列に BGM を再生する映像を作成する．

アンケートでは，今後の動画編集に VSML/VSS を使用してみたいかを「使用したくない」，「どちらともいえない」，「使用したい」の 3 段階で尋ね，VSML/VSS で動画編集を行えそうかを「作れない・作れる自信がない」，「教えてもらいながらなら作れそう」，「既存のテンプレートを改造しながらなら作れそう」，「ドキュメントを見ながらなら作れそう」，「おおよそ何も見ず一人で作れそう」の 5 段階で尋ね，さらに従来手法と比較して VSML/VSS の使いにくかった点を自由記述で尋ねた．

4.3.2　結果と考察

10 人の被験者のアンケートの回答を図 13，図 14 に示す．参加者のうち 90%が「テンプレートを元にしたりドキュメントを見たりしながらなら VSML/VSS で動画を作れそう」という意見であったため，言語としての習得難易度は高くないと考えられる．

従来手法と比べて使いにくかった点についての記述では，大きく 4 つの意見があった．表 10 にそれらをまとめている．

使いにくかった点の意見として，各オブジェクトのもつ時間長の扱い（表中の “object-length の仕様”）の理解が難しいという意見が 10 人中 4 人からあった．例えば，「`prl` タグの再生時間は，その子要素のもつ有限の再生時間のうちの最大値となる」という辺りが難しい．これらの意見から，初学者への障壁を無くすためはこの辺りの概念の整理が望ましいと分かった．

もう一つの目立った意見として，プレビュー機能の使いにくさに関する意見が多くあった（表中の “プレビュー”）．

表 10　従来と比べて使いにくかった点

object-length の仕様	プレビュー	HTML ベース	プロパティ不足
	○		
		○	
			○
	○		
○			
	○		
○		○	
○	○		
○	○		
	○		

変換器の機能として，指定時刻のフレームのプレビュー画像を生成する機能を用意していたが，プレビュー位置の指定が使い難い，プレビュー画像の生成が遅い，フレームの画像でなく区間の動画をプレビューしたい，指定したオブジェクトに関するプレビューを生成させたい，といった意見があった．動画編集を行う上でプレビュー機能は大きな役割を果たしており，その機能の拡充は今後の課題である．

その他の意見として，「HTML ベースであり事前知識がないため躓いた」や「プロパティの機能不足で実現したいことが実現できなかった」などの意見もあった．後者の意見については今後の課題である．

5. 関連研究

関連研究として，Web ユーザインタフェースライブラリ React.js の技術を使用した動画編集ライブラリである Remotion [2] がある．その特徴として，React.js をベースとしているためコンポーネント分割が可能であること，CSS のように装飾を当てられること，オブジェクト間の相対的な関係を記述できることが挙げられる．しかし，Remotion の仕様上，各オブジェクトの表示時間（フレーム）をユーザが意識する必要がある．また，Remotion は HTML レンダリングエンジンを使用し各フレームを書き出して動画を出力するため，これがボトルネックになり動画出力速度にデメリットを持つ．

XML をベースとした動画記述言語として，W3C の勧告した SMIL [3] がある．SMIL は，VSML と同様に，par 要素や seq 要素によりオブジェクト間の時間軸上の相対的な配置を記述できる．しかし，VSML とは異なり，空間的な配置についてはオブジェクト間の相対的な関係では記述しない．また，装飾と構造の分離がされておらず，装飾の情報は各オブジェクトへの参照を記述する要素の属性として指定する．よって，動画の装飾に関する推敲をするには手間がかかる．さらに，SMIL は古い規格なため（最新規格の SMIL 3.0 でも 2008 年勧告），現在の環境では再生や別動画形式へのエクスポートに難がある．

6. おわりに

本研究では，従来の GUI ベースの動画編集アプリケーションの問題点を解消するために，動画構成記述言語である VSML と VSS を設計し，それらを動画に出力する変換器を実装した．また，評価実験により，VSML/VSS によって従来の問題点を解消することができることを確認した．VSML/VSS によるテキストベースの動画記述・動画編集のメリットとして，git 等の分散バージョン管理システムによるバージョン管理や複数人による安全な同時編集の容易な実現や，生成系 AI 等の機械的手段で動画を生成する際に VSML/VSS 形式で出力させることで人手による装飾等の修正が容易にできるようになることや，VSCode 等の既存 IDE の機能（高度な文字列検索や置換，入力補完，等）を活用した効率的な動画編集の実現が考えられる．

今後の課題としては，変換器の完成度を上げるとともに，コンポーネントの時間的な移動（アニメーション）への対応が考えられる．また，動画中に複数回現れるようなコンポーネントの再利用を記述できるようにする拡張も実用上有用だと考えられる．

参考文献

[1] Cascading Style Sheet(CSS) Working Group: Cascading Style Sheets home page, https://www.w3.org/Style/CSS/.
[2] Remotion AG: Remotion, https://www.remotion.dev/.
[3] W3C: Synchronized Multimedia Integration Language (SMIL 3.0), https://www.w3.org/TR/SMIL3/.
[4] W3C XML Core Working Group: Extensible Markup Language (XML) 1.0 (Fifth Edition), https://www.w3.org/TR/xml/.
[5] W3C XML Core Working Group: XML Schema, https://www.w3.org/XML/Schema.
[6] WHATWG: HTML Living Standard, https://html.spec.whatwg.org/.
[7] YouTube: YouTube for Press, YouTube Official Blog, https://blog.youtube/press/.
[8] ＫＥＮくん：Aviutl のお部屋，https://spring-fragrance.mints.ne.jp/aviutl/.