

## 電子的映写原稿の可搬性の追求

広瀬 雄二

### 概 要

様々な文書の電子化が進められ、工学文書に限らず日常的な文書までもが計算機上で作成され電子的に保存されるようになった。最終的に印刷される文書だけでなく、近年ではスクリーンを用いて何かの発表をするための映写原稿も電子的形態のまま利用される事例が増えた。このときに一般的に利用されるのはプレゼンテーションツールという分類のソフトウェアであるが、「発表文書の映写」という目的形態の閲覧にそのソフトウェアが必要となる。そのため、当該ソフトウェアのない計算機上では再現できないことになる。決して故障率が低いとはいえないパーソナルコンピュータの世界では、機器故障が発表の大きな障害となる可能性が高い。

本稿では映写原稿が特定のソフトウェアに依存するという状況を打破することを見据え、機器やソフトウェアの依存性が少なくかつ永続的に利用できる原稿形式を考える。

### Abstract

We produce various kind of documents in the electronic format nowadays. Not only the printing documents, we recently also handle presentation sheets in the form of electronic binary data. Although most people use software, so called 'presentation tools', for developing and projection of presentation sheets, those binary data can't be handled without corresponding software. It will be a menace to a feasibility of speech when speaker's PC or software in it has some trouble.

In this paper, we propose a highly portable method and format of projection documents using HyperText Markup Language to dissolve the conditions that readability and usability of presentation documents depend on specific software, especially proprietary one.

## 1 はじめに

文書を電子化することの利点は数多い。計算機上で編集できることから執筆後の加筆修正が容易である点、印刷せずに情報伝達できることから紙資源を節約できる点、などがある。近年、映写原稿も、原稿を電子的な状態で個人用計算機（以下PC）の中に保持したまま映写機でスクリーンに投影する形式が主流となった。それ以前の映写原稿は、焼き付けスライドやOHPシートを利用するものが主流で、いずれも1枚当たりの費用が大きかったり、持ち運びや整序管理に手間がかかるという問題があった。

Mac OS用に開発されたAldus' Persuasionの登場がその問題を一気に解決した。発表の概略をPC上で入力していくと、あらかじめ選択した型枠にそった形で見栄えの良い映写原稿が一枚一枚出来上がっていく。残念ながら1997年に開発停止されたため、現在では再現することができないが、その後主流となったMicrosoft社によるPowerPointや、Apple社によるKeynoteなどがその流れを汲むこととなった。

これらのソフトウェアを使うことで、現在では形式の統一された映写原稿を、物理資源を消費することなく、かつ短時間で作成できるようになった。しかし、いくつかの問題点を抱えていることも否めない。ここでは、映写原稿を作成・投影するためのソフトウェアとして、いわゆる「プレゼンテーションツール」、つまり作文・作図をGUIで行ない、作成した原稿の映写画面生成もそのソフトウェアで行なうものを想定して議論を進める。

## 2 既存のツール利用の問題点

いわゆるプレゼンテーションツールでは、原稿に文字と図だけでなく、それらを一歩隠したり、移動したりといった「動き」を保存できる。印刷物を最終成果物と仮定する一般的な文章では、一度印刷されたものは動かないため、保存される電子的文書形式には「動き」を記録することはない。これを実現する

---

<sup>1</sup>Aldusが1994年にAdobe社に吸収され、Adobe Persuasionになった。

ため、プレゼンテーションツールでは動きを規定する書式を独自に定め、それを保存形式としている。言い換えれば、ソフトウェア固有の独自形式で保存されていることになるので、そのソフトウェアがなければ閲覧・表示することさえできない。この場合、映写原稿となるファイルだけを持っていても役に立たないことになり、使用するPCが故障した場合には中を見ることすらできなくなる可能性がある。

当該ソフトウェアを持っていない人のことを考慮して受け渡しをする場合ソフトウェアなしで見るとするには紙媒体に印刷するなど、動きのないものに落とす必要がある。この場合は発表時における「流れ」という情報が欠落したものを渡さざるを得なくなる。

以上のように、プレゼンテーションツールは原稿作成者自身の労力と出費を軽減するものの、データそのものの可搬性という観点では課題が残る。

### 3 可搬性の高い映写原稿体系の提案

既存のプレゼンテーションツールが提供する、発表の流れを表現するための「動き」を表現でき、なおかつ少しでも多くのPCで、ソフトウェアを追加することなく閲覧することのできる形式を考え、それを利用した映写原稿管理システムを提案する。

ここではまず、考慮すべき要件についてまとめる。

#### 3.1 文書保存形式の可搬性

文書を電子的に表現する場合の書式には多数のものが存在するが、

- 有償ソフトウェアを必要としない
- 数多くのプラットフォームで利用できる

という要件を満たす代表的なものを表1に示す。閲覧に際し、特殊なソフトウェアを全く必要としないプレーンテキストは可搬性の上で最も優れているが、色や文字の大きさなど、視覚的な要素を全く付加することができないので映写原稿の記述性という点では期待できない。一方、HTMLとPDFはともに視覚的効

表 1：汎用的な文書保存の書式

書式	概要
プレーンテキスト	文字そのものを対応する文字コードで保存する。どんな情報機器でも読めるため、電子メールでは原則としてプレーンテキストを使用することになっている。文字以外を含ませることはできない。
HTML	Hyper Text Markup Language；文字、画像、音声、他テキストへのリンクなどの混在した文書を交換するために設計された言語。Web ページを記述するために利用されている。
PDF	Portable Document Format；Adobe 社が汎用的な文書交換のために設計した書式。閲覧するソフトウェアが数多くのプラットフォーム用に無償配布されている。

果をも記述できるため、映写原稿の表現形式として利用できる可能性を持つ。

### 3.2 文書保存形式の作成要件

HTMLまたはPDF形式のファイルを作成するための要件を考える。

HTMLはそれ自身プレーンテキストであるため、作成のためには専用ソフトウェアを必要としない。PDFは元来Adobe社による“Acrobat”で作成するものであったが、普及に伴いPDFを生成するソフトウェアが次第に増え、別形式の文書から機械的に変換して得ることが容易になった。変換のためのツールはいくつかあるが、いずれも生成したPDFファイルを元の文書に戻すことはできない。PDF作成者とは別の人がPDFファイルを受け取った場合、それを編集するためには“Acrobat”または、専用のソフトウェアが必要となる。残念ながらPDFファイルを自由に編集できるツールで、どのプラットフォームでも無償利用可能なものは存在しない。

作成、閲覧、編集全ての面において汎用的な形式を求めるのであれば、HTMLを選択するのが有利といえる。

### 3.3 HTMLによる「動き」の表現

もともとHTMLは文書の「構造」を記述するための言語であり、HTMLソースには全て文書の論理的構造の要素を記述していく。それぞれの要素の見栄え（スタイル）をHTMLソースとは独立した形で定義するためのものがスタイル

シート[2]<sup>2</sup>である。HTML文書中に現れる全ての要素には、それを出力する媒体(画面、紙、音声読み上げ装置等)に応じたスタイルが備わっている。HTML文書中の要素、およびそのスタイルを動的に操作するインタフェース、DOM(Document Object Model) [3] を利用することでHTML文書を動的に変化させることができる。これを応用することで動きのある映写原稿をHTMLで記述できると考えられる。

### 3.4 映写原稿に求められる「動き」

プレゼンテーションツールを利用した映写原稿で表現できる動きを大別すると以下の3つに分類できる。

1. 図や画像そのものが動画として動くもの
2. 図や画像、文字がページ内を(あるいは外から)移動するもの
3. ページ内の一部のオブジェクトが隠されていて、あるタイミングで出現するもの

このうち「どれが必須か」を決める根拠として、プレゼンテーションツール登場前にも使われていた手法を考えることとする。OHPシートなどの静的原稿でも「覆いで隠す」という形で利用されてきた3は口述の流れと同調させるために利用する人が多く、ツールの新旧にかかわらず需要があるものと考えられる。よって、本稿で提案するシステムでは3の動きを実装する。この機能はHTML上でスタイルシートの「visibility属性」を利用することで実現可能である。

### 3.5 可搬性のある「動き」の実現

グラフィカルなユーザインタフェース(GUI)を持つUser Agent(以下ブラウザ)でCSSに対応したものは、軽微な差はあるが同一の画面表示属性値に対してほぼ同一の表示結果を示す。これを動的に操作するために幅広く利用できるのがJavaScript言語である。JavaScriptにはDOMを利用しHTMLの表示要素の属性を直接操作するメソッドが用意されている。

---

<sup>2</sup>Cascading Style Sheets

一定以上のシェアを持つGUIを持つブラウザはどれもJavaScriptの処理系を内包しているので、高可搬性という要求も満たしているといえる。

## 4 HTML+JavaScript による映写原稿生成システム

以下の要件を満たす映写原稿生成システムを実装し、その可用性について検証した。

1. 標準文法に則ったHTML文書がそのまま映写原稿となる(原稿作成に特別なソフトウェアを要しない)
2. 作成者にJavaScriptの知識を要求しない
3. 映写時に読みやすい大きめの文字サイズが自動的に選択される
4. 原稿の閲覧・表示がWindows/UNIX/Mac OS全てで利用できる
5. 原稿生成システム自身もWindows/UNIX/Mac OS全てで利用できる

4を実現するため、想定するブラウザを全てのプラットフォームで稼動するMozilla[4] またはその派生物Mozilla Firefoxとした。5を実現するため、全てのプラットフォームで等しく動く言語処理系で生成システム自体を記述することとし、本システムではRuby言語[5]を採用した。

### 4.1 システムの概要

Rubyを利用し、HTMLに準拠した文書を口頭発表に即した構成に変換して複数のHTML形式ファイル群を出力するシステムMoPを作成した(図1)。

生成されたHTMLファイル群にはそれぞれ、要素の表示属性等を制御する

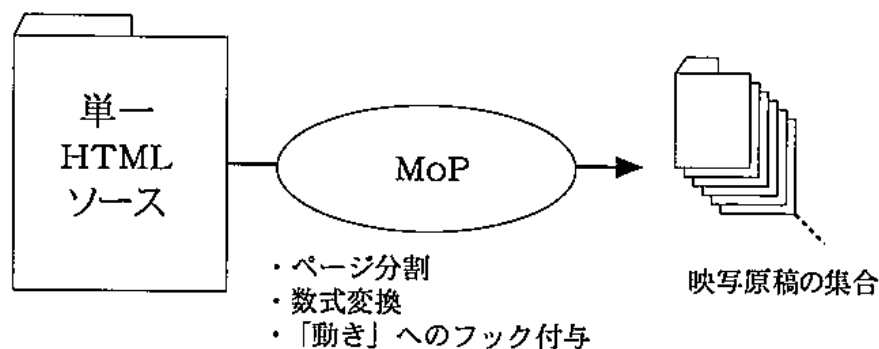


図1: MoPの動作概念

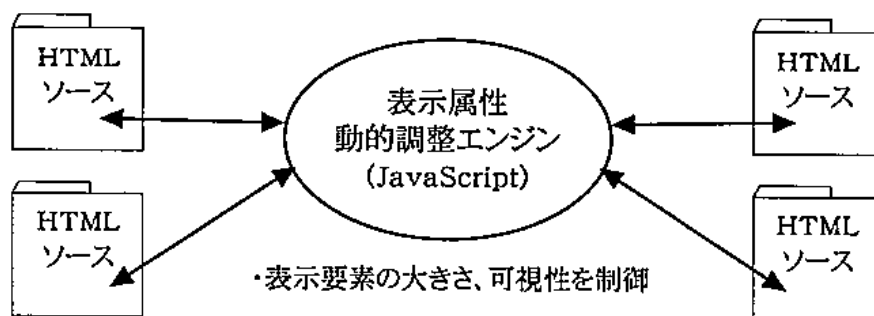


図 2：映写時に動作するMoPの動的制御系

JavaScriptプログラムがロードされ、原稿を表示しているときに閲覧者が出す指示に従って表示内容が変化する（図 2）。

## 4.2 原稿のページ分割

原稿作成者は通常のHTML文書を記述するのと同じ要領で映写原稿を記述すれば良い。通常映写原稿は聴取者が一読できる分量ごとにページ分けするが、MoPに与える原稿ではページ区切りをHTMLのh1要素で示す。本来「見出し」の意味を持つh1要素は、分割された映写原稿の見出しそのものになり、文書構造という観点での不整合はない。

## 4.3 「動き」を表すCSSクラス

原稿作成時に「動き」を持つ要素を記述したい場合は、以下に示す既定のCSSクラスを利用する。

- ・rcutinクラス、emergeクラス、opaqueクラス

要素は当初非表示。特定のキーボード（またはマウス）操作に応じて表示される。

- ・dialogクラス

クラス内部にあるblockquote要素内のパラグラフ（p要素）を、特定のキーボード（またはマウス）操作に応じて順次表示する。2者の会話を模して、交互に台詞を表示するときに利用する。

- ・qandaクラス

dialogクラスと同様。ただし、質問とその回答を摸するために質問を表示する直前には残っている文章を全て消去する。

## 5 実験

作成したシステムMoPを利用し、実際に口頭発表用の原稿を作成しプラットフォームや使用アプリケーションを選ばない可搬性が得られるかについて調べた。

### 5.1 「動き」の表現

リスト1～3に示した「動き」を示すHTMLソースをFirefoxブラウザで表示し操作したときの画面遷移を順に示す。リスト4（付録）には実験に用いたHTMLソースを掲載した。

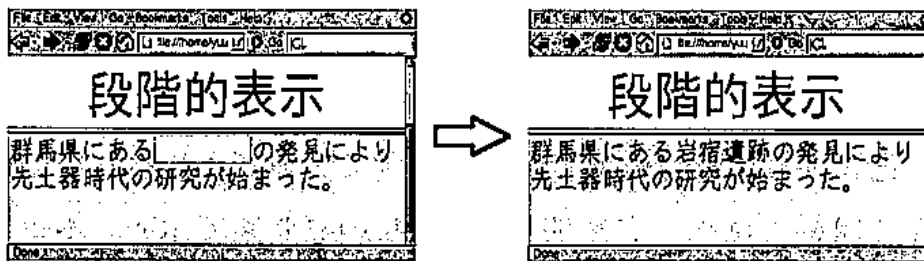


図3：emergeクラス（リスト1）指定要素の画面遷移

#### リスト1 emerge クラス

```
<h1> 段階的表示 </h1>
```

```
<p> 群馬県にある <span class="emerge"> 岩宿遺跡 </span> の発見により  
先土器時代の研究が始まった。 </p>
```

#### リスト2 dialog クラス

```
<h1> 会話表示 </h1>
```

```
<div class="dialog">
```

```
<div class="a"></div>
```

```
<div class="b" style="background: lavender;"></div>
```

```
<blockquote>
```

```
<p area="a"> ひらけ </p>
```

```
<p area="b"> ゴマ </p>
```

```
<p area="a"> ありがとうございました。 </p>
```

```
</blockquote>
```

```
</div>
```



### リスト 3 qanda クラス

```
<h1> 質疑応答表示 </h1>
<div class="qanda">
<div class="a"></div>
<div class="b" style="background: lavender;"></div>

<blockquote>
<p area="a"> 光の三原色は何ですか？ </p>
<p area="b"> 赤 (Red)、緑 (Green)、青 (Blue) です。 </p>
<p area="a"> 色の三原色は何ですか。 </p>
<p area="b"> 赤紫 (Magenta)、シアン (Cyan)、黄 (Yellow) です。 </p>
</blockquote>
</div>
```

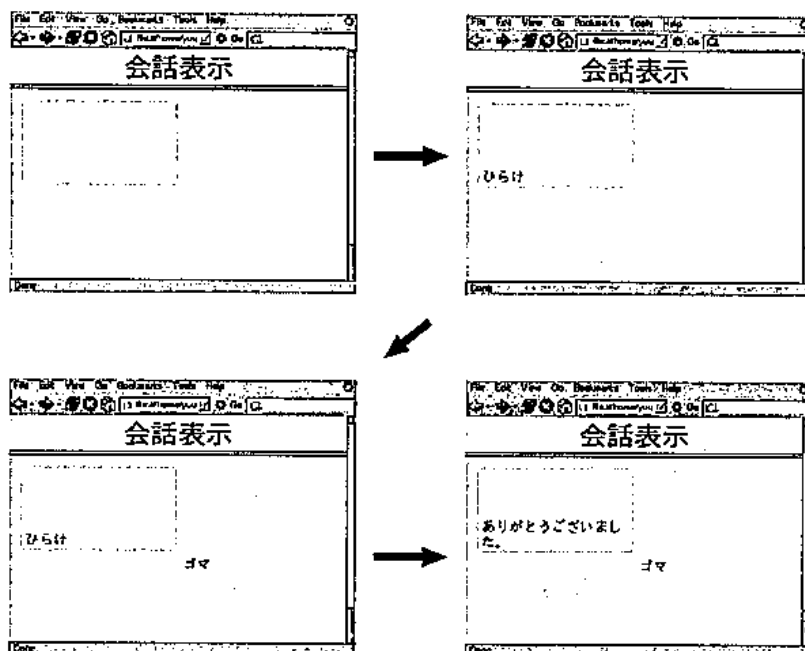


図 4 : dialogクラス (リスト 2) 指定要素の画面遷移

## 5.2 HTMLソースの可搬性の検証

JavaScriptを解釈せず、なおかつテキスト表示能力しか持たないブラウザでHTMLソースを表示した結果を図6に示す。この結果より、JavaScriptを解釈するブラウザでは「動き」を含めた内容を描写する一方、JavaScriptを解釈しないブラウザでも情報を全て表示することから、すくなくとも原稿作成者の記述した内容全てが第三者に伝わることを確認できた。リスト4を見ても明らかのように、HTMLはソースそのものを見るだけで記述内容が理解できるため、文書ファイルそのものの可搬性は高いといえる。

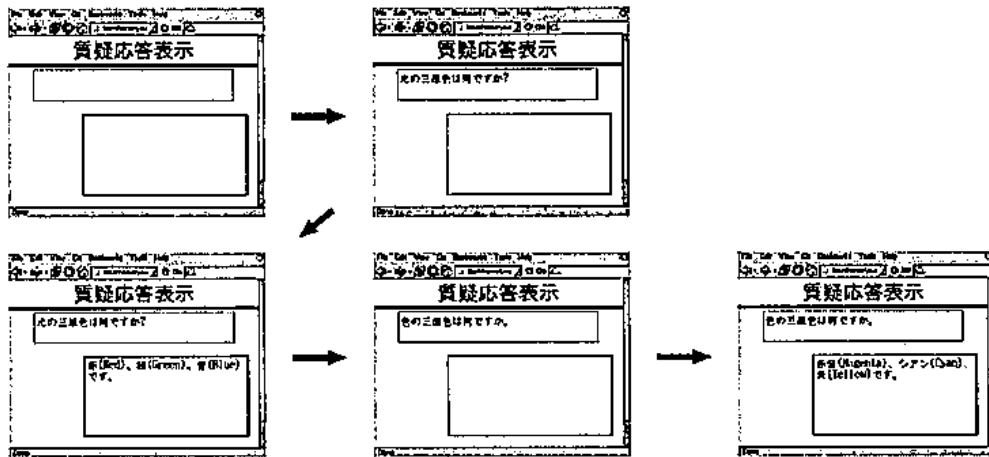


図 5 : qandaクラス (リスト 3) 指定要素の画面遷移

**段階的表示**  
群馬県にある岩宿遺跡の発見により先土器時代の研究が始まった。

**会話表示**  
ひらけ  
ゴマ  
ありがとうございました。

**質疑応答表示**  
光の三原色は何ですか？  
赤 (Red)、緑 (Green)、青 (Blue) です。  
色の三原色は何ですか。  
赤紫 (Magenta)、シアン (Cyan)、黄 (Yellow) です。

図 6 : w 3 mによる表示例

## 6 結論

本稿で提案するシステムを用いて映写原稿を作成することで、口述の流れを盛り込んだ内容を記述できると同時に、高い可搬性をもつ形式で文書を作成できる。HTMLは公共的に定められた規格であり、特定のソフトウェア提供会社の勃興の影響を受けずに永続的に利用し続けることができるのも有利な点といえる。

なお、本論で提案・作成したシステムMoPは

<http://www.gentei.org/~yuuji/software/mop/>

より入手可能である。

## 参考文献

- [1] World Wide Web Consortium; HyperText Markup Language(HTML);  
<http://www.w3.org/MarkUp/>
- [2] World Wide Web Consortium; Cascading Style Sheets;  
<http://www.w3.org/Style/CSS/>
- [3] World Wide Web Consortium; Document Object Model(DOM);  
<http://www.w3.org/DOM/>
- [4] Mozilla; mozilla.org; <http://www.mozilla.org/>
- [5] MATSUMOTO, Yukihiro; オブジェクト指向スクリプト言語Ruby;  
<http://www.ruby-lang.org/ja/>

## 付 録

### リスト4 mtest.html

```
1 <html>
2 <head>
3 <!-- MoP の JavaScript 呼出し。必ず必要。 -->
4 <script language="JavaScript" src="mop.js"></script>
5
6 <title>MoP -Template</title>
7
8 <!-- メインのスタイル定義ファイルを読む。デフォルトは basic.css -->
9 <link rel="stylesheet" type="text/css" href="basic.css">
10 </head>
11
12 <body>
13 <!-- 最初のページはタイトルページ。 mop.css の .titlepage で -->
14 <!-- スタイルが決まる。H1 で大見出し、を書く。 -->
15
16 <h1>実験</h1>
17
18 <!-- ===== -->
19 <h1>段階的表示</h1>
20
21 <p>群馬県にある<span class="emerge"> 岩宿遺跡</span>の発見により
22 先土器時代の研究が始まった。</p>
23
24 <!-- ===== -->
25 <h1>会話表示</h1>
26 <div class="dialog">
27 <div class="a"></div>
```

```
28 <div class="b" style="background: lavender;"></div>
29
30 <blockquote>
31 <p area="a">ひらけ</p>
32 <p area="b">ゴマ</p>
33 <p area="a">ありがとうございました。</p>
34 </blockquote>
35 </div>
36
37
38 <!-- ===== -->
39 <h1>質疑応答表示</h1>
40 <div class="qanda">
41 <div class="a"></div>
42 <div class="b" style="background: lavender;"></div>
43
44 <blockquote>
45 <p area="a">光の三原色は何ですか？</p>
46 <p area="b">赤 (Red)、緑 (Green)、青 (Blue) です。</p>
47 <p area="a">色の三原色は何ですか。</p>
48 <p area="b">赤紫 (Magenta)、シアン (Cyan)、黄 (Yellow) です。</p>
49 </blockquote>
50 </div>
51
52 </body>
53 </html>
```