DTPシステムにおけるフォント情報データ構造＊

4V－6

荻久保友史　栗田雅芳　今村泰介†
（株）東芝　府中工場‡

1. はじめに

ワークステーションにおけるDTP文書処理システムは、各種のウィンドウシステム環境を利用して、ディスプレイ上に印刷結果の様子をそのまま表示しながら編集できる WYSIWYG (What You See Is What You Get.) が主流である。このような文書処理システムにおいては、ユーザが使用できるフォントの種類・サイズにはかなりの選択幅が与えられており、これらの多くのフォントデータを効率よく処理することが重要となってくる。そこで本報告においては、今回開発した、ウィンドウ環境を利用した弊社文書処理システム ASDOCUMENTS[1] における、フォント情報データ構造のひとつの表現方法について述べる。

2. 文書処理システムの概要

今回開発を行なった文書処理システムは、ワークステーションのウィンドウ環境を利用し、文書編集ウィンドウ上で、テキスト、図形、イメージなど、様々なデータを混在して取り扱うことができる。ユーザは、マウスとキーボードを使用して、それらのデータを自由に編集することができる。文書編集ウィンドウ上での編集結果はそのまま、印刷結果として得ることができる。

3. フォント情報データ構造表現

まず、本文書処理システムで扱えるフォントについて概略を述べる。本文書処理システムにおいては、各種の欧字フォント、和字フォントの中から、フォントサイズ 4pt から 400pt の範囲内で 0.001pt 刻みでユーザが自由に選択して使用することができる。図1は、このフォント種類の一例である。

このようなシステムにおいては、これらフォント情報データを内部的にどのように持つかということが重要となってくる。例えば、図2のように現されているテキストを内部的にはどのようなデータ形式で現したらよいかを考えてみる。

まず考えられる表現方法としては、図3に挙げるようなものがある。データ中に含まれているIDは、それに続くテキストを表示するためのフォントの種類・文字サイズなどを指定したフォント属性テーブルを指定する。使用するフォントが変わるごとにそれに続くテキストを表示するためのフォントを現すIDを挿入する。この表現方法をデータ表現1と呼ぶことにする。

abcdefg　あいうえお
abcdefg
abcdefg　あいうえお

図1　フォント例

一方、図4に挙げるような表現方法も考えられる。和字フォントと欧字フォントはペアにして取り扱い、テキストを表示するためのフォントの種類・文字サイズなどを指定するIDは、欧字を表示するためのフォント属性テーブルと和字を表示するためのフォント属性テーブルがペアになっているテーブルを指

あいABうえCD

図2　画面表示テキスト

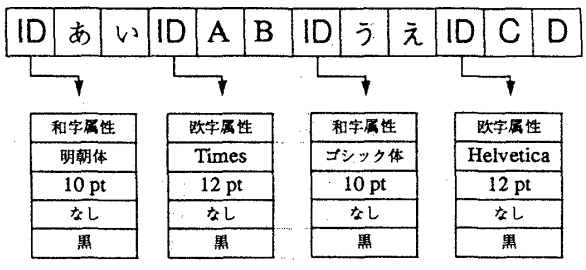

図3　データ表現1

＊ Font data structure for DTP system
† Tomofumi Ogikubo, Masayoshi Kurita, Taisuke Imamura
‡ Toshiba Corp. Fuchu works

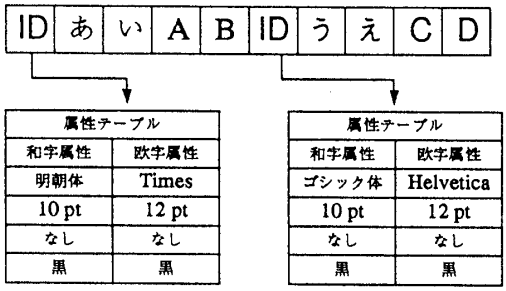

図4　データ表現2

定するようにしている。こちらの表現方法をデータ表現2と呼ぶことにする。本文書処理システムにおいては、こちらのデータ表現2の内部フォントデータ表現方法で各種処理を行なっている。

4. フォント情報データ構造の検討

4.1 データ量の検討

通常の和字・欧字が混在した文章を作成することを考えると、和字・欧字ともフォントの変更がないとき、データ表現1の場合は、文章が和字から欧字、欧字から和字に変わるごとに文字を表現するフォントを指定するためのIDの付加が必ず必要となる。その一方で、データ表現2の場合には、和字から欧字、欧字から和字に変わることがあってもフォント指定のIDを付加する必要がない。このため、データ表現1に比べ、データ量が少なくてすむことがわかる。

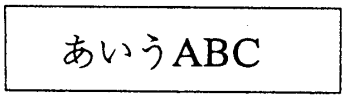

図5　画面表示テキスト（文字挿入前）

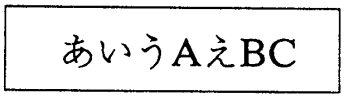

図6　画面表示テキスト（文字挿入後）

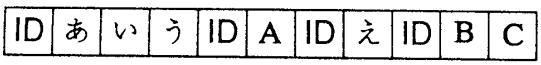

図7　データ表現1（文字挿入後）

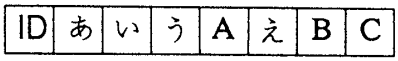

図8　データ表現2（文字挿入後）

4.2 テキスト編集処理の比較

本文書処理システムでは、データ表現2のフォントデータ表現で内部処理を行なうこととしたが、データ表現1とデータ表現2のそれぞれのデータ表現方法について、テキストへの文字挿入といった基本的なデータ処理を行なうということについて検討を行なってみる。

ここでは、欧字文字列に和字を挿入する処理について考える。図5で示してあるテキストは、和字が明朝体、欧字がTimesとなっている。このテキストの欧字「A」と欧字「B」の間に和字「え」を挿入するという処理を行なう。

データ表現1の場合、以下のような手順となる。

(1) 挿入する文字の直前に挿入する文字のフォントを指定するためのIDを付加する。
(2) 挿入文字直後のフォントを指定するためのIDを、挿入文字直後に付加する。
(3) 付加したIDとともに、挿入位置に文字を挿入する。

欧字文字列中に和字を挿入するため、文字「え」の直前に和字フォントを指定するためのIDを付加し、この和字の直後に欧字フォントを指定するIDを付加するという処理は、必須である。

一方、データ表現2の場合は、以下の手順のみである。

(1) 挿入位置に文字を挿入する。

つまり、挿入位置にそのまま文字のみを挿入する処理を行なえばよく、フォント指定を行なうためのIDに関する処理は行なわずにすむ。

この文字挿入のような基本的な処理が、データ表現2の場合には簡潔に行なうことができる。

文字挿入後の画面表示テキスト、および、それぞれのデータ表現方法の文字挿入後の様子を図6、7、8に挙げる。

5. まとめ

今回のフォント情報データ構造により、必要となるデータ量を少なくすることができ、また、テキストに対する各種処理を簡潔に効率良く行なえることを確認した。今回システムを開発していくうえでも、このような基本的な処理が簡潔になることで、プログラム設計を効率よく進めていくことができた。

参考文献

[1] 栗田他："総合OAシステムに発展するAS-Documents 文書処理基本機能 DOCMaker", 情報処理学会第41回全国大会講演論文集, 1990