文書編集時の高速表示手法

4J−4

豆田憲治　小野修一郎　吉田茂　久保登

シャープ(株)　技術本部　コンピュータシステム研究所

1. はじめに

WYSIWYGのDTPシステムでは、レイアウトされた結果を確認しながら文書の編集ができることを特徴とするため、効率的な編集作業のためには編集操作に対する素早い応答が必要となっている。しかし、多種類の書体や文字サイズ、平長体、変形文字などを扱うためにフォントをアウトライン化しており、フォントのラスターイメージ生成にかなりの時間がかかるため、結果として文字表示が遅くなり編集作業の妨げになっている。この問題を解決するために、編集途中の再表示は、できるだけフォント生成のルーチンを通らずに、編集前に既にウインドウ上に表示していたラスターイメージを転送する手法で再表示時間を短縮し、効率的な編集作業を可能とするシステムを開発した。

2. ラスターイメージ転送による再表示

2.1 再表示文字列の分類

一般に、文字列および枠の挿入、削除、サイズ変更などの文書編集により、編集対象の文字列だけでなく、編集対象より後ろの文字列についても表示位置が変わる場合がよくある。例えば、一文字入力されれば、その行のそれ以降の文字列の位置が変わり、更にその行から文字が溢れた場合には、その段落のそれ以降の文字列の位置が変わり、もし段落の行数が増加した場合には、それ以降のすべての文字列の位置が変わる。

組版処理による文字位置の変更という観点から見ると、再表示対象となる文字列について次のように分類できる。

(a) 移動しない文字列

組版の処理が行われていない行あるいは、偶然同じ位置に割り付けられた行であり、表示上は何もしなくて良い。

(b) 文書の前方に移動する文字列

行のある部分を削除したり、文字サイズを小さくしたときなどに発生する。また、枠を移動、枠サイズ変更したときにも発生する。

(c) 文書の後方に移動する文字列

行のある部分に文字列を挿入したり、文字サイズを大きくしたときなどに発生する。また、枠を移動、枠サイズ変更したときにも発生する。

(d) このページから消えた文字列

編集によって行が増えたとき、表示ページの最後付近にあった行は次のページに移動するので、表示ページから消える。また、組版処理によって自動的に挿入されていた文字(欧文のハイフンなど)が必要なくなったときなど、その文字は消える。更に、書体変更などの属性変更をしたときの編集前の文字列もページから消えた文字列として考える。

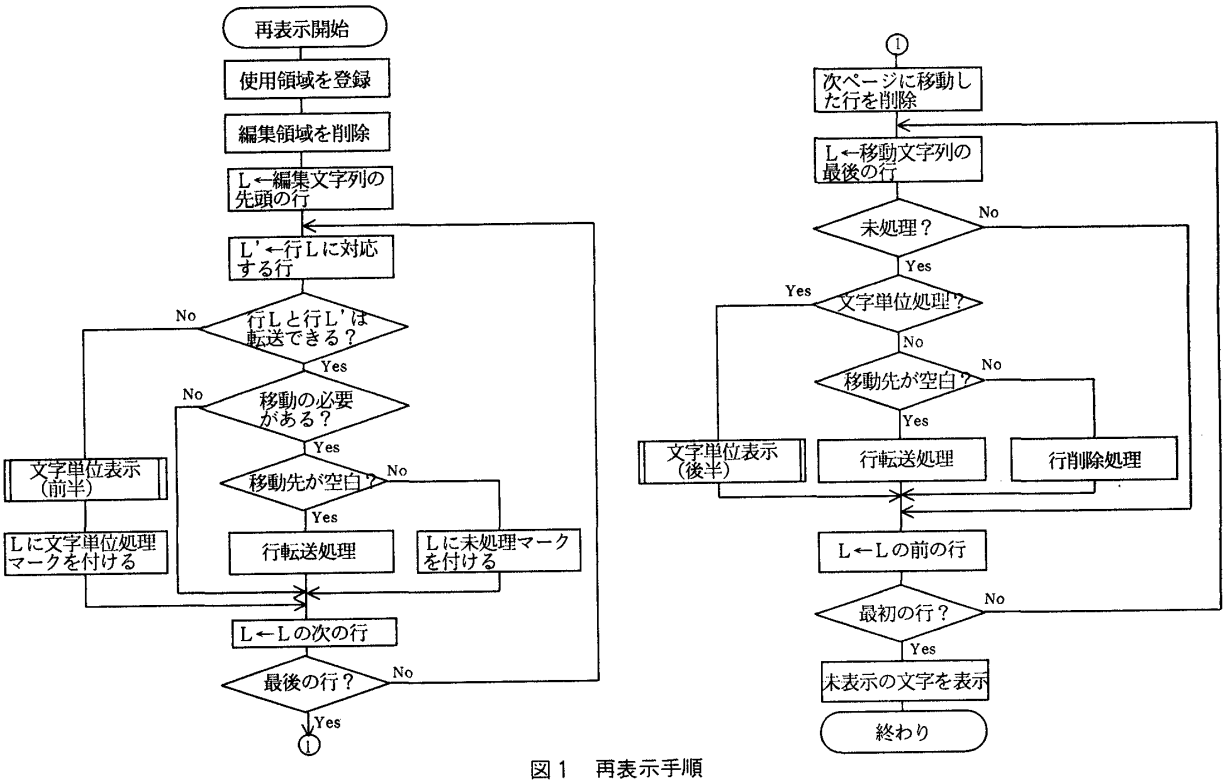

図1　再表示手順

A high speed display method on editing documents
Kenji MAMEDA, Shuichiro ONO, Shigeru YOSHIDA, Noboru KUBO
Computer Systems Laboratory, SHARP Corporation

(e) 新しく発生した文字列

編集によって行が減ったとき、ページの最後に空きができるため、次のページの先頭にあった文字列が編集ページに移動し、表示される。また、組版処理によって自動的に挿入された文字(欧文のハイフンなど)も新しく表示される。更に、書体変更などの属性変更をしたときの編集後の文字列も新しく発生した文字列と考える。

2．2　再表示に必要な情報

効率良くブロック転送を行い、再表示するためには、以下の情報および、バッファが必要である。

・編集前のレイアウト情報

・編集後のレイアウト情報

組版処理によって新しく、編集前のレイアウト情報から編集後のレイアウト情報が作成される。編集前と編集後のレイアウト情報を比較しながら、ラスターイメージの転送によって再表示を行う。

・ウインドウバッファ

文書のラスターイメージを保存しているバッファ。再表示前には編集前のレイアウト情報に対応するラスターイメージを保存しており、転送を用いる再表示はこのラスターイメージを利用して編集後のラスターイメージを作成することである。

・使用領域管理バッファ

ウインドウバッファで転送していない文字列の使用領域を記憶しておき、まだ転送していない領域への転送をしないように転送の順番を考慮する。

・転送、削除バッファ

転送（削除）は広い範囲を１度に転送（削除）した方が描画ルーチンを呼び出す回数が少なくなり、再表示時間の全体の短縮につながるので、この領域に転送（削除）すべきの文字列の情報をできるだけ広い範囲について記憶しておく。そして、記憶できなくなった時点で転送（削除）を行う。

2．3　再表示手順

図１のフローチャートの手順で再表示する。組版のレイアウト情報の１つの単位である行単位で文字列の移動方向を考慮しながら、再表示処理を実行する。

① 再表示前にウインドウバッファで使用している領域を使用領域管理バッファに登録する。

② 座標の移動が起こった行について、文書の先頭の方から順に編集後のレイアウト情報が使用領域管理バッファに登録してある領域と重なるかどうかで

(b) 文書の前方に移動する文字列

を判定し、この文字列の転送を行う。

また、編集前と編集後のレイアウト情報を比べると明らかに判定できる

(d) このページから消えた文字列

を削除する。

そして、転送、削除処理が終わると使用領域管理バッファから消去していく。

③ 座標の移動が起こった行について、②とは逆に文書の最後の方から順に編集後のレイアウト情報が使用領域管理バッファに登録してある領域と重なるかどうかで

(c) 文書の後方に移動する文字列

を判定し、この文字列の転送を行う。

そして、転送、削除処理が終わると使用領域管理バッファから消去していく。

また、転送先に未処理の文字列があり転送できない文字列はこの場面で消去しておく。この文字列は、複数の文字列が互いに位置が入れ代わるような文字列のグループの一部であるので、この文字列を削除、表示で再表示処理をすれば、残りは、転送で処理が可能となるときもある。

④ まだ、表示していない文字列について表示を行う。ここでは、

(e) 新しく発生した文字列

と、③で削除した転送できない文字列が表示される。

行単位で処理できない行は文字単位にまで細分化して行単位処理と同様に、前半で②、後半で③の手順による転送で再表示ができるかを調べる。

3．実例

図２の文書は横書き、２段組で中央に枠を配置してある。この中央の枠を２文字分左に移動する操作を行うと図３のようになる。

ここで述べた手順で再表示処理をすると図３の文書は①から㉒の領域に分かれて①から㉒の順に転送され、最後に㉓の行がフォントルーチンからデータを得て表示される。

4．おわりに

再表示を転送処理によって行うことにより、フォント生成ルーチンの使用回数を大幅に減少させ、再表示時間の短縮を達成することができた。但し、転送が遅いハードウェアではブロック転送の速度とフォントの生成する速度を考慮して、転送で処理する部分と削除、表示によって処理する部分を適切に切り分ける必要がある。

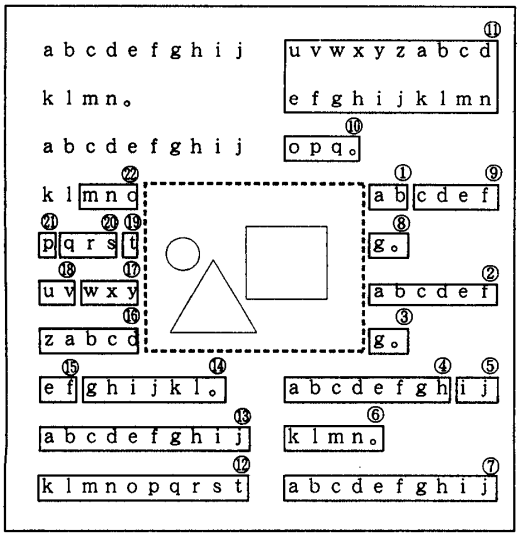

図２　編集前の表示

図３　編集後の表示