オペレーティング・システム　52－3
（１９９１．９．６）

ユーザインタフェース構築環境 鼎 (かなえ) の イメージエディタ部品

佐野 建樹

日本電気 ソフトウエア生産技術開発本部

UI構築環境鼎のイメージエディタ部品について、その構成と機能を紹介する。鼎は、X ウインドウ上のツールキットであり、イメージエディタ部品は鼎のエディタ部品の一種であり、イメージの編集機能と表示機能を持つ。編集機能には、指定領域のカット・ペースト、bitblt、カラーイメージの白黒化、イメージの拡大・縮小などがある。表示の効率化のため、遅延再表示を行う。カラーイメージを表示する際に、独自のカラー処理方式を使用している。

THE IMAGE EDITOR PART OF CANAE USER INTERFACE DEVELOPMENT ENVIRONMENT

Tateki Sano

NEC Software Engineering Development Laboratory
11-5, Shibaura 2-Chome, Minato-Ku, Tokyou 108, Japan

This paper describes mechanism and functionality of an image editor incorporated into a X window based toolkit library, "CANAE". The image editor can work on cut-and-paste over a selected region, bitblit functions, binarization, and scaling. Deferred display method has been invented to for efficient redisplay. We also introduce a color processing algorithm for color image.

1 はじめに

イメージスキャナが小型・低価格になり、イメージを蓄積するための技術、すなわち大容量の記憶装置やデータ圧縮などの技術が普及してきたため、文書にイメージデータを取り込むことが容易にできるようになった。

UNIX ワークステーションのウインドウシステムとして標準的基盤であるXウインドウシステムには、カラーイメージの編集・表示を一貫して行うライブラリが存在しない。

本稿では、カラーイメージを編集し、ユーザインタフェース構築環境 鼎 [1] の文書に取り込むためのエディタ部品であるイメージエディタの実現について述べる。

次節では、鼎の概要を説明する。3節では、鼎内でのイメージエディタ部品の位置付けについて述べる。4節では、編集対象領域のデータ構造と、マウスによる指定方法を説明する。また、5節では、表示色数に制限があるディスプレイ上で、複数のカラーイメージを矛盾なく表示するための問題点と、解決方針を示す。遅延再表示機構の中での実現方法についても触れる。

2 鼎の概要

ユーザインタフェース構築環境 鼎は、C 言語で記述された Xlib[2]、X Toolkit[3] 上のオブジェクト指向風のライブラリである。

鼎の目的は、CASE ツールなどの GUI 部分を部品化することにより、再利用率をあげ、品質を向上させることである。

鼎は、大きく分けて、対話部品とエディタ部品という 2 つの部分からなる。ここでは、エディタ部品のみ説明する。

鼎のエディタで編集できるデータは、テキスト、イメージ、図形、グラフ構造、階層構造、表の六種類である。これらをメディアと呼ぶ。イメージエディタはイメージメディアを編集する部品である。

イメージエディタは、テキストエディタと同様、他の構造エディタ (グラフ構造エディタ、階層構造エディタ、表エディタ、図形エディタ) に組み込まれ、非構造的なデータ (ラスタイメージ) を編集する部品である。

鼎エディタの特徴として、以下の点を挙げる。

メディアネスト 各メディアを相互に組み合わせ、その場で表示・編集をおこなう機能。たとえば、階層構造メディアの中に図形データを取り込み、表示し、その場で編集し、そのままファイルにセーブすることが可能である。

フラグメント メディアネストのために、フラグメントと呼ぶデータ形式を使用する。これは、マシンフラグメントは、ファイル入出力、X ウインドウのクライアント間通信にも使用される。

メディアリンク 複数のメディア間に関係を定義することができる。

エディタ部品は、メディアディスクリプタならびにビュークラスというメソッドテーブルを作成して、オブジェクト指向的にプログラムされている。

メディアディスクリプタとは、鼎のエディタで編集するデータの一般化操作に使用するメソッドテーブルである。各メディア毎に 1 つ、合計 6 つ存在する。メディアディスクリプタを使用することにより、メディア共通処理のコードを共有化でき、他のメディアを意識せずにメディアネストが出来るようになる。

メディアディスクリプタには、図 1に示すように、メディアネスト、モデルやセレクションの初期化、カット・コピー・ペーストなどのメディア依存部分が記述されている。これらは、全メディア共通のデータ操作関数から呼び出される。また、メディアネストを行う時は、暗黙に呼び出される場合がある。

ビュークラスとは、遅延再表示機構のなかからメディア依存部分を切り分けたものである。ビュークラスに入れるメディア依存機能は、図 1に

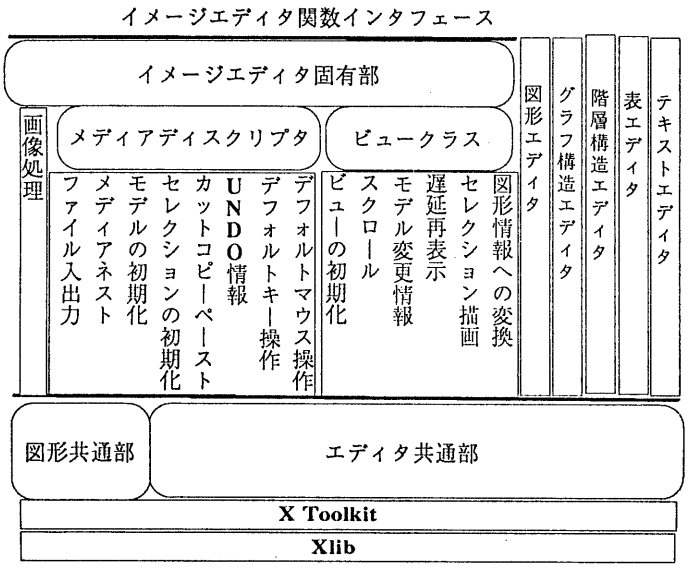

図 1: イメージエディタの実現構造

示すように、ビューの初期化、スクロール、遅延再表示などである。

メディアディスクリプタやビュークラスを利用することにより、データ抽象化と、共通部分とメディア依存部分の切りわけを行なって、保守性の向上やコードの共有による効率化を図っている。

鼎には、図形共通部とよぶ描画機能をもつライブラリが存在する。図形共通部は、固定小数点や行列の演算、図形を表現するデータ構造の操作、色やパターン、線の端点の形、線種、太さなどの属性定義、それに図形の描画機能を提供する。図形共通部で扱える図形は、点、矩形、曲線を含む折れ線、テキスト、イメージ、複合図形である。

イメージエディタでのイメージ表示と編集操作は、図形共通部により実現されている。

イメージエディタとメディアディスクリプタ、ビュークラスの関係を図 1に示す。

3 鼎イメージエディタの位置付け

イメージ編集ツールの目的は、イメージの作成と、取り込んだイメージの整形の二種類に大別できる。

イメージ作成ツールは、ペイントツールと呼ばれ、コンピュータ上で無からイメージを生成するツールである。色やパターンの設定機能のほか、マウスによるペン・ブラシ・スプレーなどの機能が必要となる。パーソナルコンピュータ上で、多くの製品が存在する。

しかし、鼎 (かなえ) イメージエディタは、イメージ作成ではなく、主にスキャナから取り込んだイメージを整形することを目的として設計した。理由は、以下の通りである。

- 作図機能は鼎の図形エディタで十分である。
- ファイル変換機能を用意すれば、他のツールで描画したイメージを取り込むことができる。
- 基本的なカットアンドペースト機能さえ実現すれば、ペイント機能は後に追加可能で

ある。

イメージスキャナから読み込んだデータ、あるいはソフトウエアで作成されたデータ、たとえばウインドウダンプ、CG などを整形するために、必要な機能は以下の通りである。

領域取り出し スキャナでは、必要な部分だけを取り出して入力することは困難である。入力時には大きめにイメージを取り込み、画面上に表示して必要な部分を切り出す方が効率的である。

画像処理 取り込んだイメージは、スキャナの性能の限界により、色合い、雑音、濃度などの調整が必要になる場合がある。

回転 スキャナで読み込む場合、イメージの方向が水平からずれる場合がある。イメージを回転させることができれば、スキャナの位置合わせが容易になり、また、位置がずれた場合に再度スキャナから入力する手間を省くことが出来る。

拡大・縮小 イメージを文書に取り込んで使用する場合、入力時のイメージの大きさが適当でない場合がある。スキャナ側で倍率を変更することも出来るが、その場合はイメージの再入力となり、また、任意の倍率変更は期待できない。

鼎イメージエディタは、Xlib、X Toolkit と図形共通部鼎を使用し、上記の機能を以下のように実現した。

イメージの変形 拡大・縮小を行う関数のインタフェースを作成した。また、マウスによるユーザインタフェースにより、イメージの拡大・縮小を行う機能を実現した (4節参照)。

画像処理 図形共通部の bitblt 演算とハーフトーン処理などを利用した関数のインタフェースを作成した。また、bitblt 関数は、カット・コピー・ペーストにも使用している。

イメージの作成 前述の理由でペイント機能は存在しない。しかし、マウスにより連続したイメージのペーストを行うユーザインタフェースを実験した。マウスドラッグごとにペーストを行うと、遅延再表示のためユーザへのフィードバックがなく、わかりにくい。また、ペースト毎に強制再表示を行うようにすると、速度に問題があった。そこで、ドラッグ時にはペースト領域の反転のみ行い、遅延再表示を利用してマウスドラッグが終了するとペースト領域全体を描画するようにした。

この方法で、ユーザへのフィードバックと速度の両方を満足している。今後、ペイント機能を作成する際に役立つと思われる。

イメージの入出力 ファイル入出力は、図形共通部の関数を利用し、いろいろな形式を 1 つの汎用関数で読み込めるようなインタフェースを作成した。現在、MacPaint、GIF、XWD、鼎形式の入出力が可能である。また、プリンタ出力用として、PostScript 形式、EPSF 形式に対応している。

なお、鼎形式は、メディアネスト機能 [4] やクライアント間通信にも使用する。

図形エディタとの連携 図形エディタは主にアウトライン図形を編集する機能をもつ。図形エディタでは、図形メディアにイメージをペーストし、その拡大・縮小や、回転、属性の変更が可能である。また、図形エディタ上でイメージエディタをネストさせることにより、その場でイメージの編集ができる。

図形エディタ内ではイメージをフラグメント形式ではなく PxRasterFormat[5] 形式で保持している。本来図形エディタ内のイメージは PxRasterFormat で表現されていたために、イメージをフラグメント形式で入力するとデータ形式が内部で 2 種類 (PxRasterFormat とフラグメント) に分離してしまう。

また、フラグメントは汎用化しているために、処理効率、速度と機能に若干の制限が存在する。したがって、そのままでは同じイメージでも内部表現により操作が異なってしまい、分かりにくくなるため、PxRaster-Format に統一した。

マウスによる操作 マウス操作には、いくつかのモードがある。セレクションの操作モードは、4節で詳しく述べる。

ピクセル検査モードは、セレクション操作とは別のモードである。このモードでは、マウスをイメージ上でクリックすると、その場所のピクセル情報が指定したコールバック関数に与えられる。

リンク検査モードは、選択操作に付加するモードである。このモードに入った後、マウスをリンク上でクリックすると、その場所のリンク情報が指定したコールバック関数に与えられる。

イメージエディタ部品を利用したプログラムの画面例を、図 2に示す。

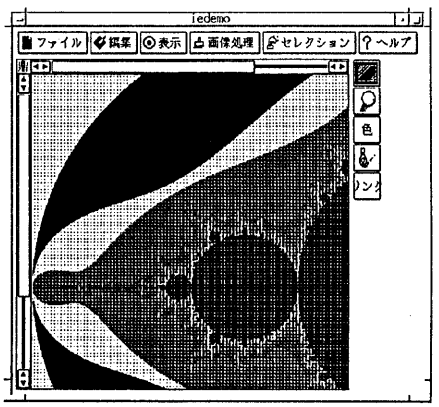

図 2: イメージエディタ画面例

4 セレクション

セレクションとは、編集モデル内の操作対象領域を表す概念である。この節では、セレクションのデータ構造と、表現方法、それに、マウスによる指定方法を述べる。

4.1 データ構造

セレクションは矩形領域とし、Xlib の Region データ構造で表現する [2]。その理由は、以下の通りである。

- スキャナから読み込んだイメージの編集では、多くの操作が矩形領域に対するもので十分であると考えられる。
- 必要があれば、一般の領域を複数矩形領域で表現することができる。その際、関数仕様の拡張が容易である。

4.2 表示方式

セレクションは、枠線により表示する。セレクションを領域の色回転や反転により表示すると、正しいイメージ表現にならない。これは、他のメディアと異なり、イメージでは色が情報の全てであるため致命的な問題となる。もとからその色だったのか、それとも選択されて反転しているのか区別がつかなくなるからである。これは、枠線を使用した場合も多かれ少なかれ起きる問題である。

これをさけるため、セレクションを動的に表示することが考えられる。しかし、X ウインドウのタイムアウトイベントによりセレクションを一定時間毎に変更すると、遅延再表示のメカニズムに影響を与えてしまうため、現在のところセレクションは静止状態で表示されている。

4.3 マウスによる設定

イメージエディタのユーザインタフェースは、最初にセレクションを設定し、その後選択対象に操作を行う、という形式である。これをマウスで操作する方法を、図 3に示す。

セレクションの変更 セレクションの枠付近で再度セレクション操作と同様にマウスをドラッ

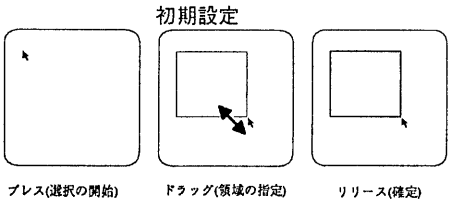

図 3: イメージエディタの選択操作

グすることにより、セレクションの変更を行う。(図 4参照)

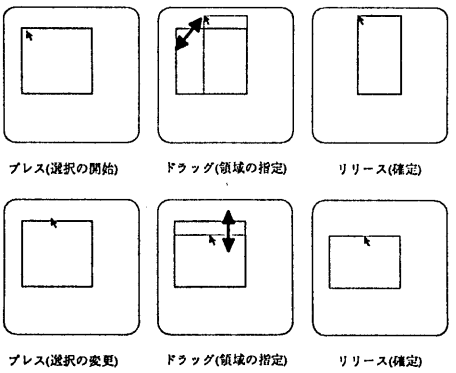

図 4: 選択の変更

この方法によると、次のような利点がある。

- セレクション変更の際の開始位置が、ノブを利用した場合に比べて広い。
- モード切り替え無しで、セレクションの新規作成と変更が可能である。
- セレクションを任意形状に拡張した場合、ノブでは表現が難しい。輪郭線ならほとんどの形状で定義可能である。
- 大きな領域を選択しても、正しく表示されない領域はわずかである。

また、欠点は、以下の通りである。

- 「枠付近」があいまいであり、セレクションの変更開始に失敗することがある。一旦失敗すると、セレクションがクリアされてしまう。
- 複雑な形状では、枠線による領域表示は分かりにくい場合がある。

セレクションの拡大・縮小 セレクションの枠付近で再度セレクション操作と同様にマウスをシフトドラッグすることにより、セレクション内部の領域の拡大・縮小を行う。

5 表示機構

この節では、カラーイメージを表示色数の少ないハードウエア上に表示するための機構について示す。また、遅延再表示のメカニズムに、それがどのように組み込まれているかを説明する。

5.1 カラーマップ

X ウインドウのサーバは、カラーマップを用いて色を表示する (図 5参照)。

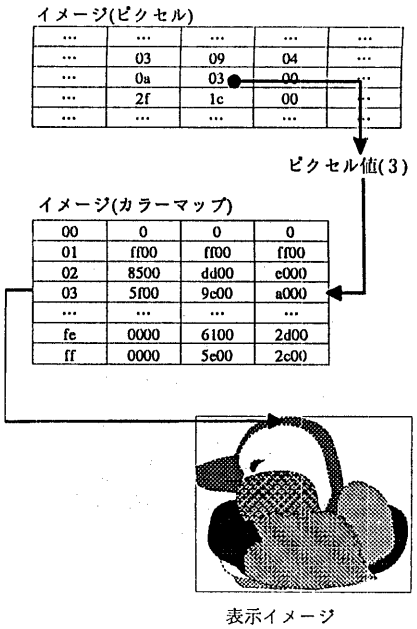

図 5: カラーマップの概念

カラールックアップテーブル (CLUT) をもつイメージを表示する場合、かならずしもイメー

ジのCLUTとウインドウのカラーマップは同じではない。

そこで、カラーマップをCLUTに一致させようとすると、以下の問題が生じる。

- カラーマップを画面全体で共有している場合は、既存のウインドウ上に表示されている色が変化してしまう。
- カラーマップをイメージ毎、あるいはウインドウ毎に持たせる場合、1画面では同時に1つのカラーマップのみ保持できる制限のあるハードウエアが多いため、前項と同様の問題が生じる。また、カラーマップの切り替え操作の方法を考える必要がある。ウインドウのフォーカスによりカラーマップを切り替えることが多い。元のカラーマップに切り替えた場合は、正常な色が表示できる。

また、イメージ側を現在のカラーマップに合うように変更するには、以下の問題点がある。

- ほとんどの場合、カラーマップに合うようにイメージを変更するための処理は、カラーマップ側を変更する処理に比べ非常に大きい。
- 現在のカラーマップに登録されていない色が存在する可能性がある。もし、登録されていない場合は、新たにアロケートするか、一番近い色を取ってくるようにすればよい。しかし、通常カラーマップのエントリ数は表示可能な色に比べて少ないため、一部の色のみ表示出来ないことがある。

イメージを単独で処理するツールであれば、カラーマップを専用化したほうが処理効率の点で有効であると思われる。しかし、鼎では、メディアネストやハイパーメディアなど、マルチウインドウ環境で動作するアプリケーションへの適用が前提である。また、クライアント間通信により、イメージデータを交換することもあり得る。したがって、鼎ではイメージ側をカラーマップに合うように変換し、表示するようにした。カラーマップにない場合は、とれる範囲で新しく色を確保していく。

5.2 遅延再表示とカラーマップの調整

鼎では、不必要な再描画を防止するため遅延再表示方式を採用している (図 6参照)。

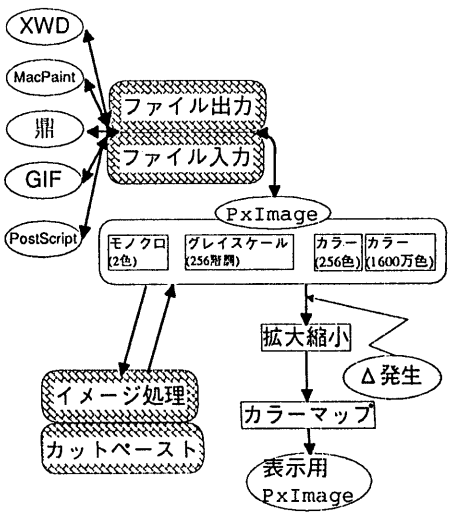

図 6: イメージの遅延表示方式

イメージエディタでは、次のような手順により再表示を行う。

1. モデルに変更があれば、変更箇所をデルタと呼ぶデータ構造により各ビューに登録する。もし、以前のデルタが存在していれば、新しいデルタを追加する。 イメージエディタの場合、イメージのピクセルの変更と、カラーマップ の変更によりデルタが発生する。
2. モデル変更が一段落すると、デルタ情報やエディタウィジットの状態の変更情報から再描画の必要な領域が計算される。デルタは、再描画必要領域の計算後、クリアされる。 なお、デルタのほかに、表示倍率の変

更によってもこの再描画必要領域 が変更される。

3. 再描画必要領域が確定すると、ビュークラスの再表示関数が呼ばれる。 再表示関数内では、再描画必要領域内のみの表示イメージを画面に転送 し、再描画する。

6 結び

本論文では、ユーザインタフェース構築環境鼎のイメージエディタ部品の実現について、セレクションの実現と、カラーマップをもった複数のカラーイメージの遅延再表示機構の実現を中心に説明した。

鼎イメージエディタのようなアウタOS的プログラムを実現する上では、ウインドウシステムやOS等に以下のような改善点を望む。

- 鼎エディタ同士のカットアンドペーストは可能であるが、プロトコルが標準化されていないため他のアプリケーションで作成したデータを取り込むことが出来ない。テキストのみではなく、図形、論理データ、その他音声や動画などもアプリケーションによらず共通化されていることが望ましい。
- 5節で述べたように、カラーマップを調節する機構がないためにイメージ表示機構が複雑になっている。ウインドウシステムに、カラーの調節機構が必要であろう。
- スキャナ入力やカラー表示などの入出力部分に関して、デバイス独立な機能が提供されていることが望ましい。

なお、イメージエディタについて以下のような改良作業を現在すすめている。

- 選択領域(現在は矩形)の一般化とマウス操作による設定。
- イメージ処理関数の高速化、効率化。
- 表示イメージのXサーバー側での保持。現在は、クライアント側に表示イメージを作成している。サーバー側で保持するようにすれば、表示を高速化できる可能性がある。
- リアルタイムの画像変換。現在、遅延再表示の際に拡大・縮小、カラーマップの調整をしているが、十分な表示速度を得ている。さらにカラーイメージのモノクロ化処理やガンマ補正等の処理を加えることが可能かどうか、検討中である。
- スキャナ入力ルーチンのライブラリ化。

謝辞

日本電気ソフトウエア生産技術開発本部の秋口忠三氏には、論文の作成をご指導頂きました。また、同部の 菅井勝氏、及び鼎グループの方々にはイメージエディタの実現に際しご指導いただきました。感謝致します。

参考文献

[1] 暦本純一, 垂水浩幸, 他: エディタを部品としたユーザインタフェース構築基盤: 鼎, 情報処理 第31巻 第5号, pp.602-611 (1990).

[2] A. Nye: "*The Definitive Guides to the X Window System, Volume 1, Xlib Programming Manual*", O'Reilly & Associates, Inc., 1988.

[3] A. Nye and T. O'Reilly: "*The Definitive Guides to the X Window System, Volume 4, X Toolkit Intrinsics Programming Manual*", O'Reilly & Associates, Inc., 1990.

[4] 暦本純一, 山崎剛, 他: 「鼎」における複合文書データ混在方式, マイクロコンピュータとワークステーション 55-2, (1989).

[5] 日本電気: EWS-UX/V EWS-UX/V(Rel 4.0) ユーザインタフェース構築環境鼎(かなえ) プログラミングの手引, 1991.