状態管理によるマルチビュー表示制御†

渡辺範人†† 福島 忠††
富田次男†† 後藤正宏†††

CAD や CAE の応用プログラムでは，ある図形の全体図と拡大図を同時に表示するマルチビューと呼ばれる機能が，マンマシン・インタフェース上欠かせないものとなっている．しかし，マルチビューでは，ビューの個数の増大に伴い，描画する図形量が飛躍的に増加するため，表示更新に時間を要し，応答性が悪くなるという欠点がある．この問題を解決するため，本論文では，①ビューの表示更新を活性，不活性，不可視の状態により管理する，②グラフィックス関数を，自動的に再表示を行うべき影響範囲により分類する，③システムコールの単位で表示更新を一括処理する，方式を提案している．

1. はじめに

グラフィック・ハードウェア技術の発展に伴い，グラフィックを利用する応用ソフトウェアは大量の図形データを扱い，表示処理システムに対し種々の機能を要求するようになった．特に，CAD (Computer Aided Design) などの応用ソフトウェアにおいては，使い勝手を良くするために機能が豊富であること，また，大量の図形データを高速に表示できることなどが重要となる．なかでも，応用ソフトウェアのマンマシン性を向上させるものとして，マルチビューと呼ばれる機能がある[1]．これは，同一図形データを同時に複数の倍率や見方で表示する機能であり，これにより応用ソフトウェアは**図1**に示すように，図面の全体図や，拡大図などを同時に表示することができるようになる．なお，それぞれの表示領域はビューと呼ばれる．

マルチビュー機能を利用すると，ユーザは画面から，図形データを把握しやすくなるという利点はあるが，その反面，表示する図形量が増え，表示更新の時間が長くかかってしまうという欠点も生じる．例えば，マルチビューにおいて，図形データを変更すると，通常はそのデータを表示しているすべてのビュー内の図形が変更されるのが前提である．しかしながら，応用ソフトウェアによっては，速度や，使い勝手の点から変更前の図形も残しておきたいといった要望もある．従来のマルチビュー機能では，一般に表示更新はすべてのビューに対して行うか，または，一つのビューのみについて行うかの選択しか許されず，応用ソフトウェアが自由に表示を制御できなかった．このため，図形データの変更がすべてのビューの表示更新を誘起してしまって，いつも長い時間表示がちらつくことになったり，一つのビューしか表示が更新されず，結局すべてのビューを表示し直さないと希望のビューを表示更新できないといった不都合が生じていた．特に，CAD などでは，使い勝手，および応答性は，マンマシン性を大きく左右するものである．マルチビュー機能においては，表示をどの時点で，どのビューに対して行うかを応用ソフトウェアが制御できることが重要となる．つまり，表示を更新する際には，画面のちらつきが少ないこと，高速に行えることが大切である．これらの観点に立って，基本ソフトウェアにマルチビュー機能を備えたエンジニアリング・ワークステーションを開発した．

本論文では，ユーザ・インタフェース向上を目的として，①ビュー状態による表示制御，②関数分類による表示更新処理の最適化，③表示の一括更新による処理オーバヘッドの削減，を図ったマルチビューの一実現方法について論じる．

2. システム概要

2.1 システム構成

本システムのハードウェア構成を**図2**に，ソフトウェア構成を**図3**に示す．同図に示すように，本システムは，オフィス向けワークステーション[2]をグラフィックス機能について拡張したものである．

本システムにおいては，図形の拡大・縮小を行ったり，視点を変えたりすることによる，図形の描き直しや，図形のピック処理を高速に行うため，図2に示す

† A Multi-View Display Method with the State Control by NORITO WATANABE, TADASI FUKUSIMA, TSUGIO TOMITA (Hitachi Research Laboratoy, Hitachi, Ltd.) and MASAHIRO GOTO (Omika Works, Hitachi, Ltd.).
†† (株)日立製作所日立研究所
††† (株)日立製作所大みか工場

ようなハードウェア構成をとっている[3]．本システムのハードウェアは，CPU (Central Processing Unit), SB (Segment Buffer), GDP (Graphic Display Processor) などから構成されている．図形を描画する場合は，CPU が GDP の図形描画命令を SB 上に作成し，GDP に描画開始を指示する．GDP は，CPU の指示に従って SB 上の図形描画命令を実行し，フレーム・メモリ上に図形を展開する．この結果，図形がグラフィック・ディスプレイ上に表示される．

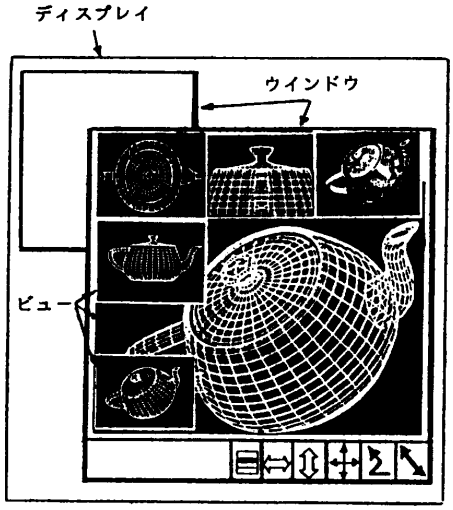
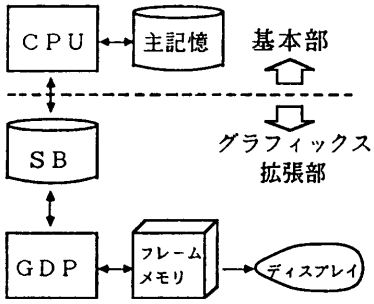


図 1 マルチビュー表示例
Fig. 1 Example of multi-view.



SB: Segment Buffer
GDP: Graphic Display Processor
図 2 ハードウェア構成
Fig. 2 Hardware configuration.

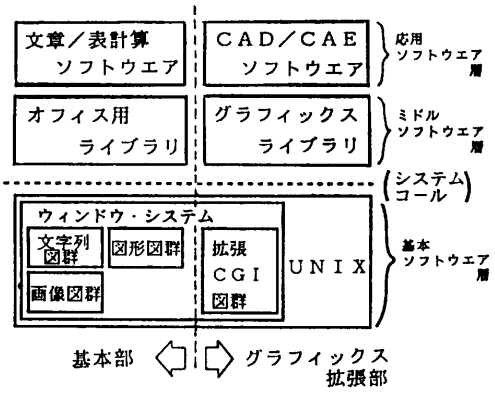


図 3 ソフトウェア構成
Fig. 3 Software structure.

一方，ソフトウェアは，図3に示すように，基本ソフトウェア層，ミドル・ソフトウェア層，および，応用ソフトウェア層から成る[4]．基本ソフトウェア層は，UNIX* System V をベースとしたオペレーティング・システムである．本システムではマルチウィンドウ機能は，このオペレーティング・システムのカーネル内に構築されている．本システムでは，ウィンドウ内で，文字や，イメージ，図形等を統一的に扱って，文章を作成できる，マルチメディア対応のアーキテクチャをとっている．つまり，各メディアのデータを処理するための，専用の基本ソフトウェアをウィンドウ・システム内に組み込み，各基本ソフトウェアが各々のデータの表示，および，管理を行うようになっている．これらの基本ソフトウェアを，本システムでは，図群と呼んでいる[2]．例えば，文書処理を行う文字列図群や，イメージを処理する画像図群などがある．今回付加したグラフィックス機能についても，文書中に CAD などで作成した図面をはめ込むといったことを可能にするため，図群（以下，拡張 CGI 図群と呼ぶ）として，ウィンドウ・システム内で実現することにした．一方，ミドル・ソフトウェア層は，GKS (Graphical Kernel System)[5] などのグラフィックス・ライブラリを構築している．応用ソフトウェアは，これらのライブラリを介して，拡張 CGI 図群のグラフィックス機能にアクセスすることになる．

2.2 拡張 CGI 図群

拡張 CGI 図群は，表示を行う図形の描画命令を SB 上に作成し，GDP の描画起動を制御する．機能的には，ISO (International Organization for Standardization：国際標準化機構）において標準化が進められている CGI (Computer Graphics Interface)[6] の機能をベースに，マルチビュー機能をはじめ，多階層セグメント・データ，セグメント・クラス，3次元プリミティブやシェーディングなど，大幅に機能を拡張したものである．

マルチビューを実現するにあたり，決定しておかな

* UNIX は米国 AT & T 社ベル研究所が開発したオペレーティング・システムである．

ければいけないものに，座標変換，遅延制御，表示更新制御がある．以下に，これらについて説明する．

（1） 座 標 変 換

座標変換とは，応用ソフトウェアが定義した図形の座標値に，どのような変換を作用させ，画面上にマッピングするかを決定するものである．拡張 CGI 図群では，3次元図形データを扱えるように標準グラフィックス・インタフェース（案）である PHIGS (Programmer's Hierachical Interactive Graphics System)[7] を参考にして，座標変換系を決定することにした．しかし，PHIGS の座標変換系では，図形データ中に，ビューイングの切替え命令が定義されるため，ビューイングの変更を行おうとする際には，すべてのビューについて，描き直しを行う必要がある．また，図形データの途中でビューが切り替わるため，ビューのオーバラッピング制御が難しい，といった問題がある．そこで，拡張 CGI 図群では，図形データと，ビューイングの管理を分離し，ビューイングの変更，表示の更新をビュー独立に行えるようにした．さらに，出力した図形が，上に重なっているビューにはみ出さないよう，オーバラッピング制御を行った．

拡張 CGI 図群の座標変換系は，**図4**に示すような5段階の座標変換から成る．応用ソフトウェアが定義した図形は，MC (Modelling Coordinates) 空間上に，部品として定義され，モデリング変換により，WC (World Coordinates) 空間で一つの図面として組み上げられる．次に，ビュー方向変換により，視点を基準とした座標空間 VRC (View Reference Coordinates) に変換後，ビューの配置を決定する座標空間 VDC (Virtual Device Coordinates) に変換される．さらに，ワークステーション変換により，図群ごとの座標空間である図群 DC (Device Coordinates) 空間に変換される．最後に，マルチウィンドウを実現するためのマルチウィンドウ変換が施されて，実際の画面へマッピングされる．（なお，マルチウィンドウ変換は，すべての図群共通の機能であり，ウィンドウ・システムが提供している．）

上記座標変換のうち，マルチビューは，WC→VRC→VDC の変換（ビューイング変換と呼ぶ）を用いて実現した．ビューイング変換は複数の変換定義を許し，WC 上に定義された図形は，画面の複数の位置に出力される．ここで，VDC 空間にマッピングされる領域（VDC ビューポート）が，それぞれ一つのビューを形成することになる．

（2） 関数バッファリング制御

拡張 CGI 図群は，図3に示すように，オペレーティング・システムのカーネル内にある．拡張 CGI 図群に対するグラフィック関数は，UNIX の WRITE システム・コールにより，エスケープ・シーケンス形式のデータとしてカーネル内に取り込まれる．応用ソフトウェアが発行した個々のグラフィックス関数を，個別のシステム・コールとしてカーネルへ送れば，直ちに実行されて，表示に反映される．しかしながら，一般にシステム・コールのオーバヘッドは大きいた

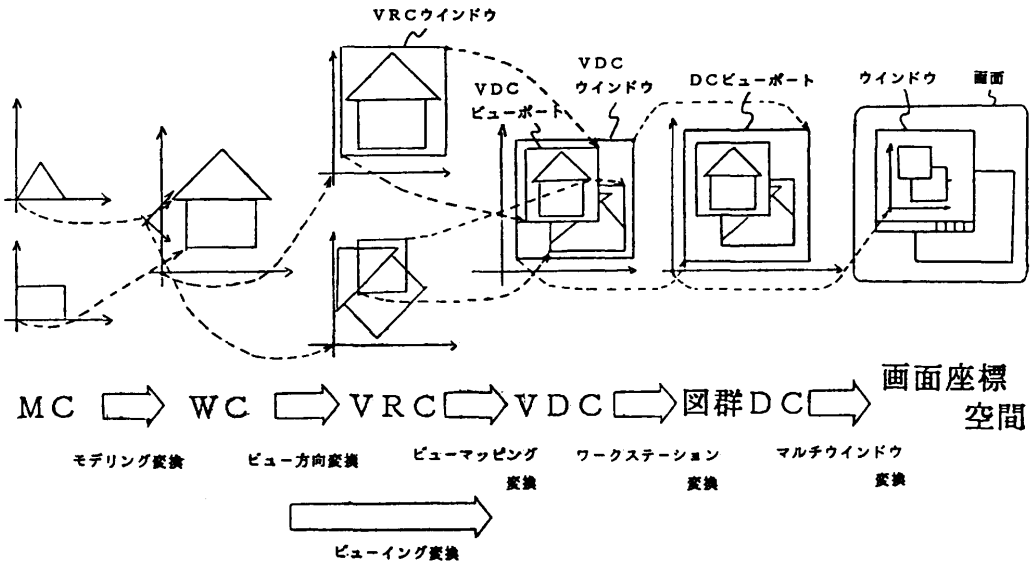


図 4 座 標 変 換
Fig. 4 Viewing pipeline.

め，システム・コールの回数の増加に伴って，処理性能が低下する．このオーバヘッドを低減し，図形の高速表示を実現するため，本システムでは，視覚上の効果に影響しない範囲内で，グラフィックス関数をバッファリングする方式を採用した．なお，このバッファリングは，UNIX の標準入出力関数により実現し，図面を逐次的に更新するか，バッファリングしてから表示更新するかは，応用ソフトウェアが制御できるようにした．

（3） 表示更新制御

図形データを編集する場合，修正する度に，その結果を画面上で確認したいとか，修正結果は，後で一度に見たいなど応用ソフトウェアによってさまざまな要求がある．これに答えるため，拡張 CGI 図群では，図形データを編集したときに自動的に行う表示更新に対して，実行するか否かを制御する手段を応用ソフトウェアに提供した[8]．また，実行しない場合に対しては，応用ソフトウェアが手動的に表示更新できるようにした．

自動的に表示更新を行う方法については，図形データや，管理テーブルの変更の種類によって，次の二つに分類して，行うことにした[5)~7)]．

（a） lead to an implicit regeneration (IRG)

（b） can be performed immediately (IMM)

（a）は，表示面を一度クリアして，すべての図形データを再描画しなければ修正できない再表示タイプの表示更新であり，（b）は，クリアしなくても，即座に（部分的に）表示を更新できる部分修正タイプの表示更新である．

なお，以下本論文では，自動的に表示更新を行う場合に焦点をあて，その実行制御方式について論ずる．

3. ビュー状態管理

マルチビューでは，ビューは複数同時に表示することができるが，応用ソフトウェアによっては，ポップアップ・メニューなどのように，必要となったときにだけビューを表示したい場合がある．また，入力エコーを特定ビューに対してだけ出力し，他のビューには表示したくない，といった要求もある．このように，自動的な表示更新が常にすべてのビューに行われるのは，ユーザ・インタフェース上必ずしも好ましいとは言えない．

従来は，上記のような表示処理を行う場合，ビューの生成，削除と，自動的な表示更新の実行，非実行および手動による表示更新の組合せで対応せざるをえなかった．ポップアップ・メニューのような単純な表示処理ならば，ユーザ・インタフェース上問題は少なかったが，大量の図形データを扱う場合は，処理に時間がかかる再表示が随所で実行されることになり，使い勝手および作業効率の上から大きな障害となる．

そこで，拡張 CGI 図群では，ユーザ・インタフェースの向上を図るため，ビューごとに表示更新の制御を受けるか否かの状態を管理することにより，使いやすいインタフェースを構築することにした．まず，ビューを，表示を行う可視ビュー (visible view) と，表示を行わない不可視ビュー (invisible view) に区分した．これにより，不可視から可視，可視から不可視へビューの状態を変更することで，ポップアップ・メニューなどを容易に作成できるようになる．

さらに，可視ビューについては，自動的に表示更新を行う活性ビュー (active view) と，行わない不活性ビュー (deactive view) に分離した．つまり，図形データに対して削除や追加を行ったとき活性ビュー内の図形は操作に対して即座に反応し，不活性ビューに対してはそのままの状態となる．これにより，不要な表示更新処理を抑え，必要なビューのみを表示更新で

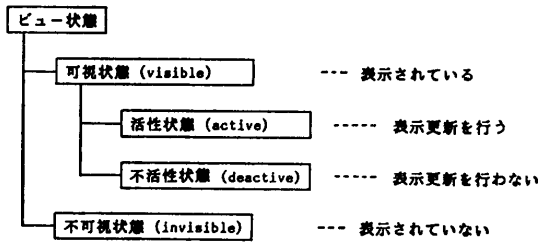


図 5 ビュー状態
Fig. 5 View states.

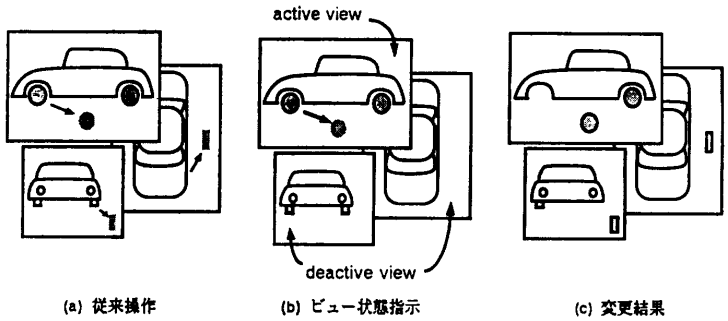


図 6 ビュー状態使用例
Fig. 6 Example of view state.

きる．ビュー状態と表示更新の関係を，**図5**に示す．

例えば，**図6**(a)に示すような図形に対し，タイヤを移動させる場合を考えてみる．このときすべてのビューの図形を逐次更新していくことは処理時間の点で効率が良くない，またちらつきが多く発生する．そこで図6(b)に示すように，現在対象としているビューのみを活性状態にしておくことにより，不要なビューの表示更新を抑え対話性を高めることができる．また，タイヤの位置が決まった時点ですべてのビューに対し最新状態にする指示をすることにより図6(c)のような表示を行える．このような操作はCADなどの応用ソフトでは頻繁に起きることから，本手法は操作性を大きく向上させるものとなる．

4. 表示更新処理

4.1 表示更新方法による関数分類

表示更新を起こす関数として再表示タイプと部分修正タイプの表示関数があることは前述した．さらに，再表示を促す関数に関しては，表示更新がどこまで影響するかは，各関数の機能により定まる．例えば，応用ソフトウェアがあるビューの背景色を変更しようとする場合，そのビューに対して再表示処理を行う必要があるが，他のビューに対しては，表示更新を行う必要はない．また，ビューの位置を移動する場合は，移動前と移動後の両方の部分を再表示しなければならない．そこで，再表示タイプの自動的な表示更新関数を，表示更新が作用する範囲に従って，次の二つに分類し，異なった処理方式を採ることにより，効率的な表示更新を行うようにした．

(1) 特定ビューに対して再表示を行う関数
(2) 図群全体に対し再表示を行う関数

(1)は，SET VIEW BACKGROUND COLOR（ビュー背景色の変更）のように，ビューを特定して，そのビューの属性や座標変換を変更するものである．このような関数には，指定ビューのみを再表示すればよいものと，SET VDC VIEWPORT（ビューの移動）のような，他のビューにまで表示更新が及ぶものがある．一方，(2)は，SET BUNDLE REPRESENTATION（バンドルテーブルの変更）のように，ビューを特定しない再表示タイプの表示更新関数である．これには，活性ビューにのみ影響が及ぶものと，SET DC VIEWPORT（ワークステーション変換の変更）のように，活性，不活性を含めすべてのビューを再表示する必要のある関数とがある．

(1)のような再表示タイプの関数を，ビューIRG関数，(2)のような再表示タイプの関数を，図群IRG関数と呼び，各関数分類に応じ必要となる部分のみについて表示更新を行うことにした．

4.2 表示更新処理

表示の更新処理を効率的に行うために要求されるのは，次の2点である．

(1) 表示更新を行うための必要な情報を，即座にとり出す．
(2) 表示更新を行うために描画する図形量を減らす．

以下，この2点を中心に，表示更新の処理方式を説明する．

4.2.1 管理構造

拡張CGI図群が管理しているデータには，大きく分けて，図形データと，管理テーブルがある．ここでいう図形データとは，GDPの図形描画命令であり，図2のSB内に格納される．図形を描画する場合，CPUがGDPに対して図形描画命令の開始位置を指示することにより，GDPは描画を開始する．一方，管理テーブルは，GDPに対し描画指示を行う際に，必要となる情報を格納するためのものである．表示更新に必要な情報としては，各ビューごとの座標変換パラメータや，背景色，さらに，ビューのオーバラッピングを行うためのビュー可視矩形情報などがある．ビュー可視矩形とは，実際に表示されるビューの領域を，矩形に分割したものであり，表示を行うとき他のビューに覆われている領域をビュー可視矩形により除くことで，ビューのオーバラッピングを実現している．

これらの情報を，拡張CGI図群では，図群全体に対し作用するものと，ビュー個別に作用するものとに分け，それぞれ，拡張CGI図群管理テーブル，および，ビュー管理テーブルと呼ぶテーブル内に格納することにした．**図7**に管理テーブルの構成概略図を示す．

図形の描画を行う際には，GDPに対して，座標変換パラメータや図形描画命令の開始位置などを指示する必要がある．このため，拡張CGI図群管理テーブルと，出力を行おうとするビューに対応するビュー管理テーブルとを検索し，必要なデータを得なければならない．しかし，描画を行う度に，毎回各ビューの管理テーブルを捜し，ビュー状態をチェックすることは，効率が良くない．また，ビュー可視矩形を求める

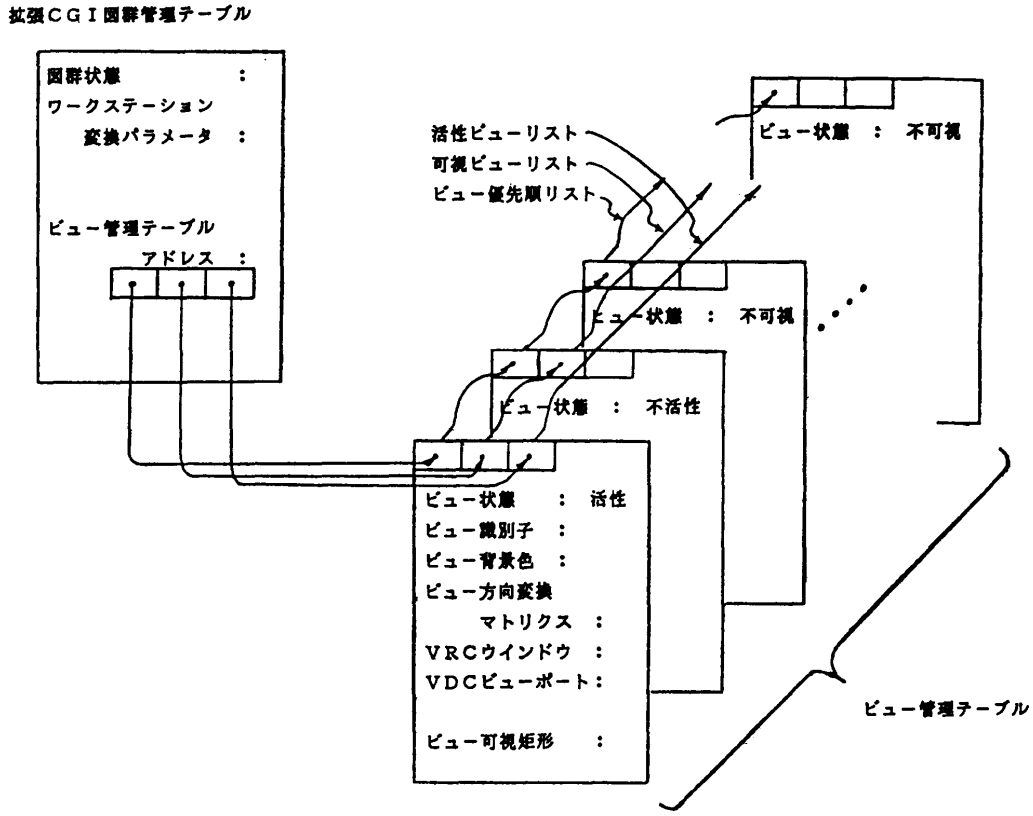


図 7 テーブル構成
Fig. 7 View management tables.

ためには，各ビューの表示優先順位を効率良く知る必要がある．そこで，図7に示すように，各ビュー管理テーブルを，次の3本の優先順リストを用いてポインタ結合することにより，検索の高速化を達成した．

① ビュー優先順リスト：すべてのビュー管理テーブルを表示の優先順に連結

② 可視ビュー優先順リスト：活性と不活性のビューの管理テーブルを表示の優先順に連結

③ 活性ビュー優先順リスト：活性ビューの管理テーブルのみを優先順に連結

以上3本のビュー優先順リストを，描画や，可視矩形計算などの処理で使い分けることにより，ビュー管理テーブルの効率的なサーチを可能とした．

4.2.2 一括表示更新

2.2 節で述べたように，拡張 CGI 図群に対するグラフィックス関数は，複数関数がまとまって送り込まれる．このとき，送り込まれた関数内に，再表示を促す関数が複数存在する場合，これらの関数を受け付ける度に再表示を行うのでは，画面がちらつく上，途中の再表示は，無駄になる．例えば，まず，ビューを移動し，さらに内部の図形を拡大する場合を考える．これら一連の処理は，VDC ビューポートの変更と，VRC ウィンドウの変更により実現されるが，このとき，変更関数ごとにそれぞれの処理を行うと，まずビューを移動するために再表示が行われ，さらに，拡大のため再び再表示が行われることになる．このように，再表示が2度続けて行われるのは，応答性の上からも，見やすさの点からも好ましくない．そこで，拡張 CGI 図群では，システム・コールの終了時点で，一括して表示更新処理を行い，ユーザ・インタフェースの向上を図ることにした．ここで，問題となるのが，次の3点である．

(1) 自動的な再表示を実行するか否かのチェック．

(2) 変更要求のあったデータの管理．

(3) ビュー IRG 関数で，他のビューにまで作用を及ぼす処理の扱い．

(1)は，WRITE システム・コール終了時に，一括して表示更新を行うため，表示更新関数の発行の有無を常にチェックする必要があることから生じる．この問題に対しては，IRG フラグと呼ぶ再表示タイプの

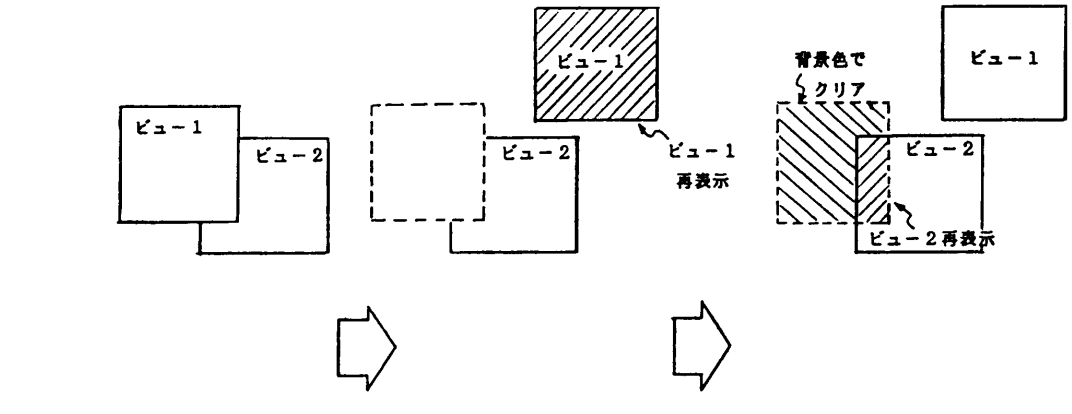
表示更新要求があったか否かを示すフラグを持つことにより対処した．

また，自動的に再表示を行う場合には，発行された関数が，図群 IRG 関数か，ビュー IRG 関数かにより，適切な表示更新の方法を決定する必要がある．そこで，図群管理テーブルに IRG 図群変更フラグを，各ビュー管理テーブルに IRG ビュー変更フラグをそれぞれ持つことにした．各変更フラグには，変更関数に対応するビットを設け，IRG 図群変更フラグは，図群 IRG 関数が発行された時点で，IRG ビュー変更フラグは，ビュー IRG 関数が発行された時点で，各関数の対応するビットを ON にすることにした．これにより，システム・コール終了時にどのような再表示を行う必要があるかを，容易に判定できるようにした．

次に，(2)の変更データの管理については，関数を受け付けた時点で，単にビュー管理テーブルの内容を変更し，システム・コールの終了時に再表示を行う場合を考えてみる．このとき，システム・コールの処理が完全に終了する前にウィンドウ操作が発生すると，部分的な再表示処理が実行され，一部分だけ変更要求のあった最新の状態の図形が描画されてしまう．また，ビューを移動する場合，前にどこにビューがあったかが分からなくなるため，常に，全面の再表示を行わなければならなくなる．これらの問題を解決するために，各ビュー管理テーブルのビュー属性や，図群管理テーブルの座標変換パラメータなどについては，現在画面上に出力されている情報（カレント値）と，変更要求された情報（リクエスト値）を持つことにした．システムコール終了時に再表示を行う場合は，リクエスト値の有無をチェックして，もしリクエスト値があるならば，その値を用いて再表示処理を行うことにした．

次に，(3)は，ビューの移動や状態の変更などの際に，そのビューに覆われていた部分の表示をいかに行うかである．この場合，影響が及んだビューは，そのビューに対して変更要求があったのではないことから，ビュー全面に対して再表示を行う必要はない．そこで，IRG ビュー変更フラグに，特別にビュー可視矩形変更ビットを持ち，このビットにより，部分的な再表示が必要なことを下位ビューに対し知らせることにした．つまり，あるビューの更新処理が，下位ビューに対し影響を及ぼす場合，影響が及んだビューを捜し，可視矩形変更ビットを ON にする．次に，下位ビューについて更新処理を行う際，このビットが ON ならば，新しく表示されることになった領域を求め，その領域のみについて再表示を行うことにした．

ここで，図 8 に示すように，ビュー 1 とビュー 2 が重なっているとする．このとき，ビュー 1 を移動させる場合を考える．この場合，表示更新処理は，以下の



	ビュー1	ビュー2
VDCビューポート変更ビット	ON	OFF
ビュー可視矩形変更ビット	OFF	OFF

(a)

	ビュー1	ビュー2
	OFF	OFF
	OFF	ON

(b)

	ビュー1	ビュー2
	OFF	OFF
	OFF	OFF

(c)

図 8 ビュー移動処理
Fig. 8 Example of moving view process.

ようになる.

① ビュー1を移動する関数 SET VDC VIEWPORT を受け付けた時点で，IRG フラグと，ビュー1の IRG ビュー変更フラグの VDC ビューポート変更ビットを ON にする (図8(a)).

② WRITE システム・コールの終了時点で，IRG フラグにより，表示更新要求の有無をチェックする．要求ありの場合は，IRG 図群変更フラグをチェックした後，IRG ビュー変更フラグを，全可視ビューについて，上位ビューから順にチェックし，表示更新を行う対象を決定する.

③ ビュー1の処理の際，IRG ビュー変更フラグの VDC ビューポート変更ビットが ON となっているので，ビュー1の再描画を要求のあった位置で行う．さらに，下位表示優先順位のビューで，ビュー1に重なっているビューを可視ビュー優先順リストから探索し，ビュー2のビュー可視矩形変更フラグを ON にする (図8(b)).

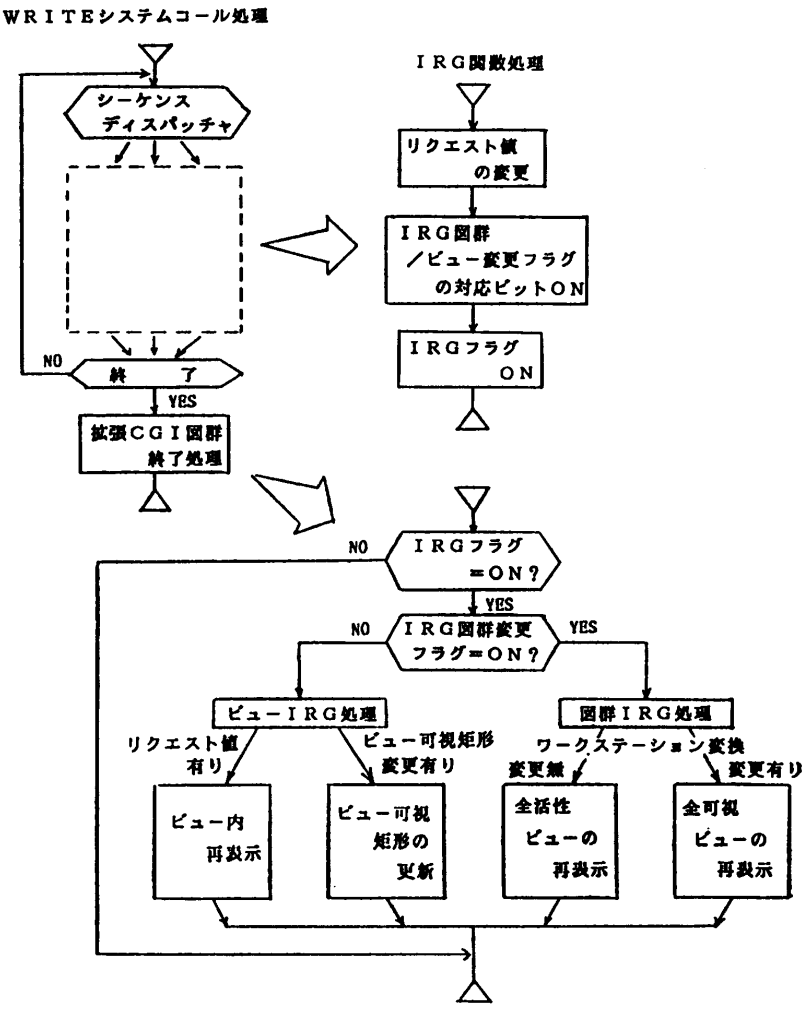


図 9 表示更新処理概略フロー
Fig. 9 Flow of implicit regeneration process.

④ ビュー2の処理の際，ビュー可視矩形変更ビットが ON となっているので，新しいビュー可視矩形を求める．新旧の可視矩形から，ビュー2の新しく表示されることになった領域を求め，その領域の再表示を行う (図8(c)).

以上の処理を行うことによって，必要な部分のみの表示更新でビューの移動が可能となる.

図9に，IRG フラグ，IRG 図群変更フラグ，IRG ビュー変更フラグの利用方法を，概略の処理フローで示す.

5. おわりに

エンジニアリング・ワークステーションのグラフィックス基本ソフトウェアにおいて，自動的な表示更新処理を効率良く行うマルチビュー制御方式を開発した.

本方式では，まず，ビューに活性，不活性，不可視の状態を与えた．そして，自動的な表示更新を行う関数を，その影響する範囲から分類し描画する図形量が，最小になるように制御した．さらに，表示更新を一括に行うことによって，無駄な表示更新処理を省くことで高速化を達成した.

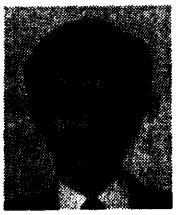
今後は，3次元図形処理の立場から，さらに使いやすいシステムを目指したい.

謝辞 本研究を御支援下さった(株)日立製作所大みか工場情報端末設計部保田勲部長，同社システム開発研究所第3部福岡和彦主任研究員，およびソフトウェア開発に御協力頂いた諸氏に深く感謝する.

参考文献

1) Foley, J. D. and Dan, A. V.: *Fundamentals of Interactive Computer Graphics*, The System Programming Series, Addison-Wesley (1982).
2) 小島ほか：マルチウィンドウ・システムの開発思想，第 32 回情報処理学会全国大会論文集，pp. 1755-1756 (1986).
3) 石井ほか：エンジニアリング・ワークステーション—開発思想—，第 36 回情報処理学会全国大会論文集，pp. 669-670 (1988).
4) 新納ほか：エンジニアリング・ワークステーション—ソフト・アーキテクチャー，第 36 回情報処理学会全国大会論文集，pp. 677-678 (1988).
5) International Standard ISO 7942, Information Processing Systems—Computer Graphics—Graphical Kernel System (GKS) Functional Description, ISO (1985).
6) Working Document, Computer Graphics Interface (CGI) (Revision 3—CG-VDI—Baseline Document) : ANSI X 3 H 3, ANSI (1985).
7) Draft dpANS PHIGS, Information Processing Systems—Computer Graphics—Programmer's Hierarchical Interactive Graphics System, ANSI (1985).
8) 福島ほか：エンジニアリング・ワークステーション—表示制御—，第 36 回情報処理学会全国大会論文集，pp. 679-680 (1988).

(昭和 63 年 7 月 18 日受付)
(平成 元 年 10 月 11 日採録)



渡辺 範人（正会員）

1960 年生．1983 年九州大学工学部情報工学科卒業．1985 年同大学院総合理工学研究科情報システム学専攻課程修了．同年(株)日立製作所日立研究所に入社，エンジニアリングワークステーションのグラフィックス基本ソフトウェア，ユーザインタフェースの研究・開発に従事．ACM 会員．

福島 忠（正会員）

昭和 28 年生．昭和 52 年京都大学工学部電気第 2 工学科卒業．昭和 54 年米国イリノイ大学コンピュータサイエンス学部修士課程修了．昭和 55 年(株)日立製作所日立研究所入所，現在に至る．画像処理，エンジニアリングワークステーション，マイクロプロセッサ，グラフィックスなどの研究に従事．京都大学工学博士．IAPR TC 6 委員．IEEE 会員．

富田 次男（正会員）

1950 年生．1973 年茨城大学工学部電子工学科卒業．同年(株)日立製作所日立研究所に入社．プロセスコンピュータのデータ管理，ラボラトリオートメーションなどの研究・開発を経て，現在グラフィックス基本ソフトウェアおよびユーザインタフェースの研究・開発に従事．

後藤 正宏（正会員）

1955年生．1978 年早稲田大学理工学部電気工学科卒業．同年(株)日立製作所大みか工場に入社．エンジニアリングワークステーションの OS，基本ソフトウェアの開発に従事．グラフィックスインタフェース，アーキテクチャに興味を持つ．