マイクロコンピュータとワークステーション50 － 3
（１９８８．５．１３）

状態管理によるマルチビュー表示制御方式

†渡辺　範人　　†福島　忠　　†富田　次男　　‡後藤　正宏

†(株)日立製作所日立研究所　　‡(株)日立製作所大みか工場

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。

A MULTI-VIEW DISPLAY METHOD WITH THE STATE CONTROL

Norito Watanabe†, Tadashi Fukushima†, Tsugio Tomita† and Masahiro Goto‡

†Hitachi Research Laboratory, Hitachi Ltd.
4026 Kuji-cho, Hitachi, Ibaraki, 319-12 Japan
‡Omika Works, Hitachi Ltd.
5-2-1 Omika-cho, Hitachi, Ibaraki, 319-12 Japan

This paper describes a method of the implicit display control in implementing the overlapping multi-view which displays a designed object at several different places on a screen with different view angles. A CAD engineer as a user of a graphic workstation strongly desires such a function as to promote his design environment. In fact, the multi-view function provides the designer with a better man-machine interfacefrom a viewpoint of observation, while it raises the increase of the entire drawn figure volume. To solve this problem we developed the implicit display control method which is featured by ①implicitly regenerating each viewport in keeping with its view state: active, deactive and invisible, ②classifying implicit regeneration functions according to their influence extent, and ③merging several regeneration executions requested in a system call.

1．はじめに

CADなどのグラフィックスを利用する応用ソフトウエアにおいては、マルチビューと呼ばれる、ユーザ・インタフェース上欠かせない機能がある。これは、図1に示すように、図面の全体図や、拡大図など複数の図を同時に表示する機能であり、それぞれの表示領域はビューと呼ばれる。このマルチビュー機能の使い勝手、及び応答性は、大きくマンマシン性を左右するものである。特に、表示をどの時点で、どのビューに対して行なうかを応用ソフトウエアが制御できることが重要である。さらに、表示を更新する際には画面のちらつきのないこと、高速に行われることも大切である。これらの観点に立って、基本ソフトウエ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、マルチビュー機能のユーザ・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ーヘッドの削減、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エア構成を図2に、ソフトウエア構成を図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[1]をグラフィックス機能について拡張したものである。

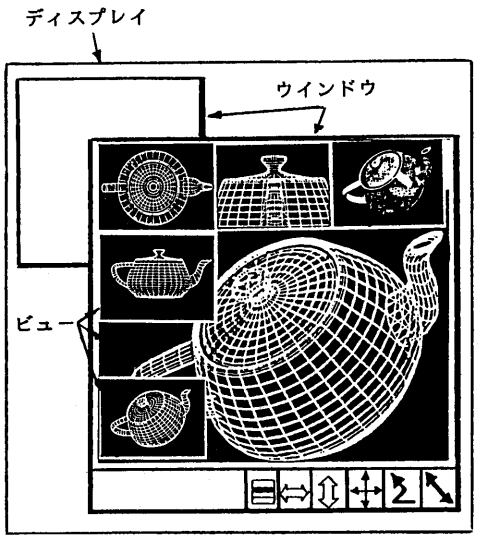

図1　マルチビュー表示例

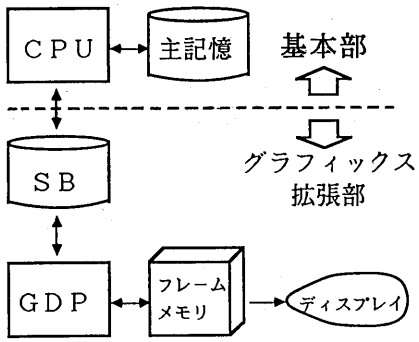

図2　ハードウエア構成

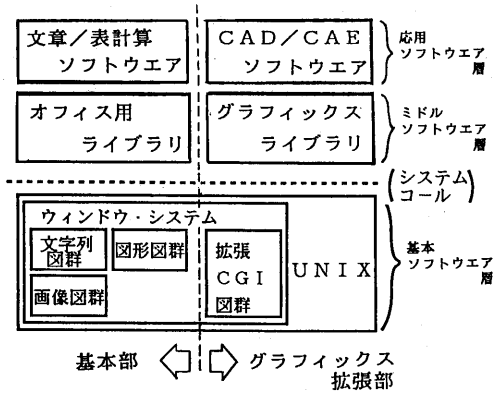

図3　ソフトウエア構成

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に示すようなハードウエア構成をとっている[2]。本システムのハードウエアは、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ことになる。

一方、ソフトウエアは、図3に示すように、基本ソフトウエア層、ミドル・ソフトウエア層、及び、応用ソフトウエア層から成る[3]。基本ソフトウエア層は、UNIX System V リリース2.0[注1]をベースとした

注1）UNIXは米国AT&T社ベル研究所が開発したオペレーティンング・システムである。

オペレーティング・システムである。マルチウインドウ・システムは、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。本システムのマルチウインドウ・システムでは、ウインドウ内で、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エアをウインドウ・システム内に組み込んでいる。これらの基本ソフトウエアを、本システムでは、図群と呼んでいる[1]。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、マルチメディア対応に、拡張ＣＧＩ図群として、ウインドウ・システム内で実現することにした。一方、ミドル・ソフトウエア層は、ＧＫＳ(Graphical Kernel System)[4]などのグラフィックス・ライブラリを構築している。応用ソフトウエアは、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)[5]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、応用ソフトウエ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)[6]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、ビューイングの切り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、全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かわるため、ビューのオーバー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。応用ソフトウエ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

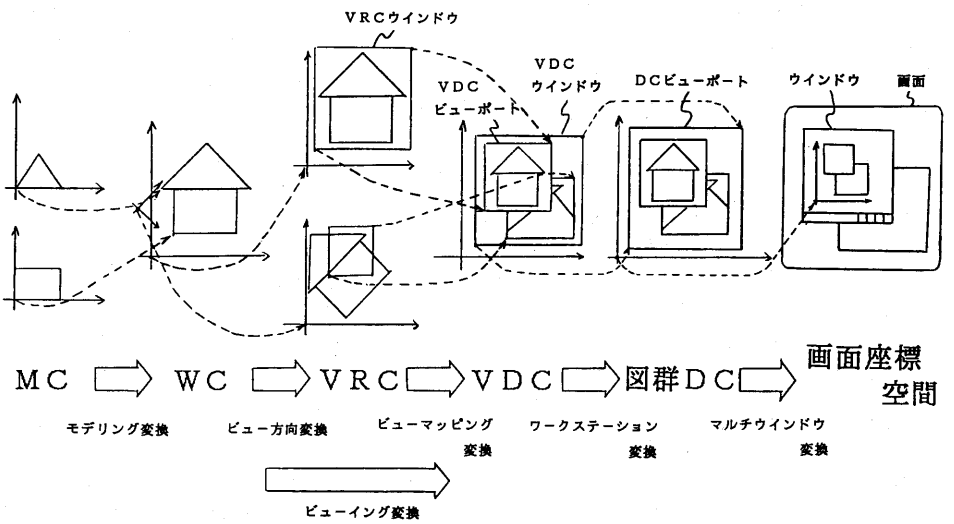

図４　座標変換

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、図群毎の座標空間である図群ＤＣ(Device Coordinates)空間に変換される。最後に、マルチウインドウを実現するためのマルチウイ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、マルチウインドウ変換は、全ての図群共通の機能であり、ウインドウ・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すことから、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とになる。つまり、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、カーネル内にあり、拡張ＣＧＩ図群関数は、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。応用ソフトウエ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・コールのオーバーヘッドは大きいため、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。このオーバーヘッドを低減し、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、応用ソフトウエ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エアによって様々な要求がある。これに答えるため、拡張ＣＧＩ図群では、表示更新を自動的に実行するか否かを制御する手段を応用ソフトウエアに提供した[7]。

また、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[4) 5) 6)]。

(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、応用ソフトウエアによっては、ポップアップ・メニューなどのように、必要となった時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全てのビューに行われるのは、ユーザ・インタフェース上好ましいとは言えない。

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、ビュー毎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、表示更新を行う方式をとることにした。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した。

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 （acvite view）と、行わない不活性ビュー（deactive view）に分離した。これにより、応用ソフトウエアは、自動的に表示更新されるビューを任意に決めることができるようになった。

ビュー状態と、表示更新の関係を、図５に示す。

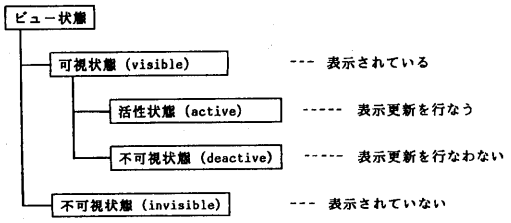

図５　ビュー状態

４．表示更新処理

４．１．　表示更新の分類

応用ソフトウエ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自動的な再表示処理を行う必要があるが、他のビューに対してはビュー状態にかかわらず不要である。また、ビューの移動の

場合は、移動前と移動後の両方の部分の表示が必要となる。このように、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。そこで、効率的に表示更新を行うため、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、異なった表示更新を行うことにした。

(1) 特定ビューに対し再表示を行う関数

(2) 図郡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、不活性を含め全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、図郡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図郡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、必要となる情報を管理するためのものである。表示更新に必要な情報としては、各ビュー毎の座標変換パラメータや、背景色、さらに、ビューのオーバー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る。他のビューに覆われている領域をビュー可視矩形から

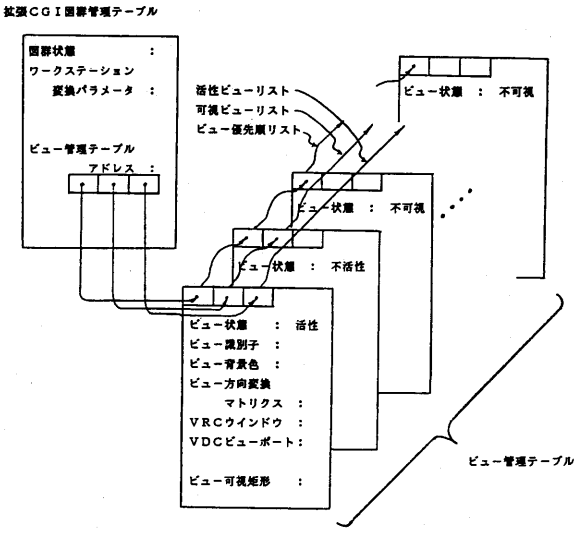

図６　テーブル構成

除くことで、ビューのオーバーラッピングを実現している。

これらの情報を、拡張ＣＧＩ図郡では、図郡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図郡管理テーブル、及び、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６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図郡管理テーブルと、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、効率がよくない。また、ビュー可視矩形を求めるためには、各ビューの表示優先順位を知る必要がある。そこで、図６に示すように、各ビュー管理テーブルを、次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：
　全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表示の優先順に連結

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．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。この時、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ウインドウの変更により実現されるが、この時、変更関数毎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を行い、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、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、関数を受付た時点で、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。この時、システム・コールの処理が完全に終了する前にウイ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有るならば、その値を用いて再表示処理を行うことにした。

次に、(3)の問題は、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

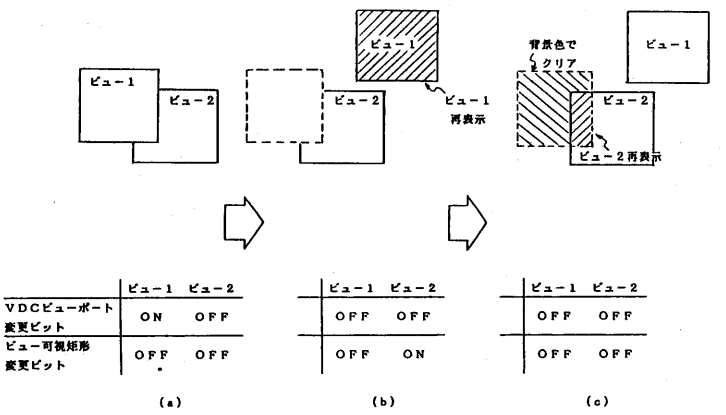

図７　ビュー移動処理

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事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７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。この時、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７（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。要求有りの場合は、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７（ｂ））。

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７（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８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。

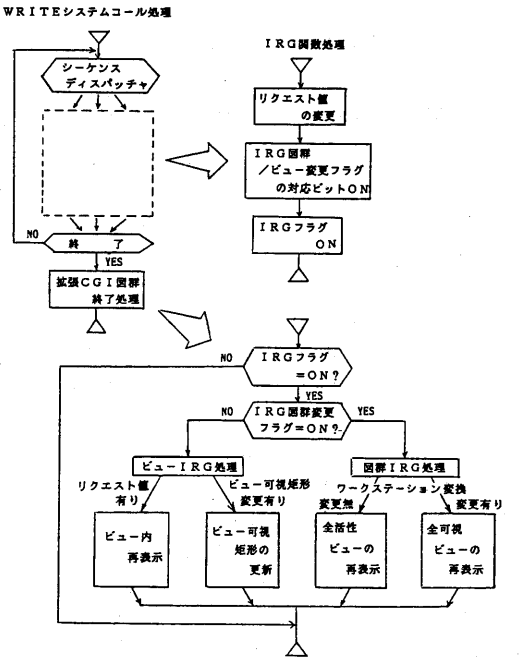

図８　表示更新処理概略フロー

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エアにおいて、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保田部長、同社システム開発研究所福岡主任研究員、及び御協力頂いた諸氏に深く感謝する。

[参考文献]
1)小島他：マルチウインドウ・システムの開発思想，情報処理学会第３２回全国大会(1986,3)
2)石井他：エンジニアリング・ワークステーション－開発思想－，情報処理学会第３６回全国大会(1988,3)
3)新納他：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，情報処理学会第３６回全国大会(1988,3)
4)International Standard ISO7942, Information Processing Systems -Computer Graphics- Graphical Kernel System (GKS) Functional Description (1985,8)
5)Working Document, Computer Graphics Interface (CGI) (Revision 3 -CG-VDI- Baseline Document) :ANSI X3H3 (1985,3)
6)Draft dpANS PHIGS,Information Processing Systems -Computer Graphics- Programmer's Hierarchical Interactive Graphics System (1985,8)
7)福島他：エンジニアリング・ワークステーション－表示制御－，情報処理学会第３６回全国大会(1988,3)