コンピュータ・グラフィックス*〔2〕

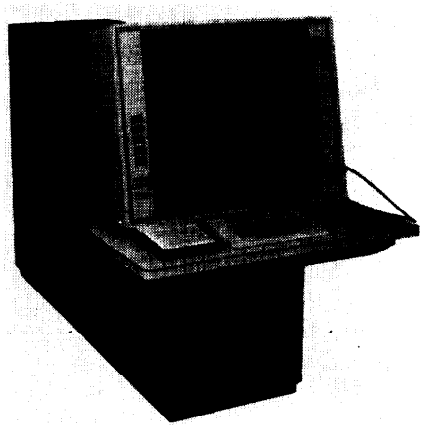
大須賀節雄**

3.3 グラフィック・システムの構成[31]

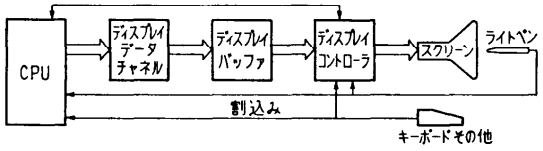
これまでグラフィック・システムの基本となる各種の装置について述べてきた（Vol. 12, No. 4, pp. 229～239 参照）．図形という視覚に直接に訴える媒体を介して，人間と計算機が対話できるシステムを構成する要素として，これらの新しいハードウェアが不可欠であることはいうまでもないが，その機能を活用するためには適切なシステムを構成すること，および対話型の利用を可能にするソフトウェア・システムの開発が重要である．

コンピュータ・グラフィック・システムが大きな期待をかけられているにもかかわらず，予想外にその普及が遅れている最大の原因はコスト高にある．ここでコストとは絶対額としてのコストおよびその有用性との比較における実質コスト（コスト・エフェクティブネス）の両方を意味する．これらの解決をはかるためには，まず第一に各装置の価格を下げることが重要であるが，さらに，システム全体としてコストを下げるため，適切なシステム構成への配慮が必要であり，またコスト・エフェクティブネスを改善するには，有用なソフトウェアを開発してゆくことが重要になるのである．この節では，まずシステム構成から述べることにしよう．

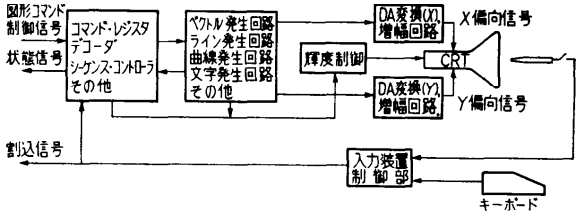
汎用計算機とグラフィック・ターミナル(**第7図**)とから成るグラフィック・システムの標準構成は**第8図**のように表わされる．すなわち，グラフィック・ターミナルとしてはディスプレイ・データ・チャネル，ディスプレイ・コントローラ(**第9図**)，ディスプレイ・スクリーン，各種入力装置などが基本ユニットであるが，通常はこれにリフレッシング用のバッファメモリを加えたものから成っている．この構成では入力装置からのオペレータの指示により，計算機のプログラムが実行され，表示用の図形コマンドの列（ディスプレイ・ファイル）がディスプレイ・チャネルを通してディスプレイ・バッファに記憶される．ディスプレイ・コントローラはこれを順次読み出してスクリーン上に表示する．専用ディスプレイ・バッファを置かず，計算機の主メモリの一部を利用するメモリ・サイクル・スチール方式のものもある（**第10図**）．



第7図



第8図



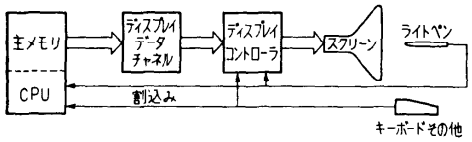
第9図

このようなシステムでは，ターミナル・コストに加えてそれを制御するための大型計算機のコストが大きくなり，グラフィック・システムが経済的な意味で妥当な手段になりにくい．特に，インタラクティブであ

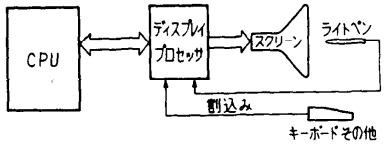
---

* Computer Graphics [2], by Setsuo Osuga (Institute of Space and Aeronautical Science, University of Tokyo)

** 東京大学宇宙航空研究所



第 10 図



第 11 図

ることはオペレータの意志のもとに計算機が作動することであるから，計算機の方がしばしばオペレータの次の指示を待っということも生じてくる．このような状況は必ずしもグラフィック・システムに限られたものでなく．むしろ，インタラクティブ・システムに共通の問題であるが，特にグラフィック・ディスプレイ・システムにおいてそれが大きな問題になるのは，図形というものが非常に多量の情報を含むものであり，それをコミュニケーションの手段として用いるシステムには短時間できわめて多量の情報交換能力が要求されるという点にある．端末の装置の情報交換能力は十分大きいので，これと同程度の情報処理能力を計算機に課することは，計算機にとってかなりのロードとなる．

このような理由から時分割方式を用いて，マルチ・コンソールにするということだけでは必ずしも満足な解決策にならない．インタラクティブであることを重視し，端末装置の能力を活用することを心がけるなら，計算機に接続しうるコンソール数はたかだか数台の程度であろう．反対に計算機側の経済性を優先してコンソール数を増したとすると，各コンソールでの待ち時間が増し，もはやインタラクティブとはいえなくなってくる．

このような状況のもとで，最近の傾向として現われてきたのがマルチプロセッサ方式である．グラフィック・システムにおいては多レベル，多種類のプログラムが準備されていて，オペレータの指示によりそれらが実行されるが，そのうちのかなりの部分は画面上の図形の修正や削除，挿入，変換などにかかわるものである．図形の変換（移動，回転，拡大・縮小，透視）などにはかなりの演算時間を要し，さらに画面の変更ごとに，新しい図形要素を発生してその図形コマンドを作り，ディスプレイ・ファイルを修正したり，作り換えたりする必要がある．これらの仕事を大型の計算機から切り離し，小型の専用の計算機に任せることにすれば，1コンソールあたりに大型計算機の行なう処理および待ちの時間は大幅に減少するであろう．そして各コンソールのユーザには妥当な応答時間を保証し，かつほかのコンソールのユーザにあまり影響を与えることなく，コンソール数を増すこともできる．このような端末は情報処理能力を有するという意味においてIntelligent-Terminal とも呼ばれるが，これを図示したのが**第 11 図**である．この図で小型計算機(ディスプレイ・プロセッサ）が図の表示やユーザとのインタラクションを行ない，さらに通信線または I/O チャネルを通して大型計算機に結合されている．大型計算機はユーザの問題に対応する大きなデータ構造を保有し，またその高速の処理能力を用いてアプリケーション・プログラムを実行する．しかしながら，実際にはこの両計算機間のプログラム配分の問題はかなりむずかしく，これがこのシステムの一つのポイントになっている．一般に両計算機間にかなり大量の情報伝送が要求されるが，これを極力少なくし，かつレスポンスを阻害しないようなプログラム配分を，小型計算機における容量や処理能力の限度内で行なわれなければならない．この問題はソフトウェア・システムのあり方とも関連するので，まだ十分解決しているとはいえない．しかし，いずれにしろグラフィック・システムにおいて，多数のコンソールが同時に複雑な CAD の問題を扱えるようにするには，この小型計算機利用の方式がおそらく唯一の方式であろうとされている．

3.4 グラフィックス・ソフトウェア

グラフィック・システムのように，コミュニケーションの媒体として図形を利用するようになると，従来計算機利用の範囲外にあった広い分野の仕事がこの範囲内に組み入れられるようになり，同時に計算機利用の方法にも本質的な変化が生じてきた．元来図形というものは人間向きの問題表現形式であるから，それが直接計算機に入力されるということは，問題そのものが計算機に与えられることを意味する．従来は問題を解く手順が与えられていたのであるから，このことは計算機にとって非常に大きな変化である．このように図形処理とは Problem Solving と呼ばれる種類の問題の一種ともいえるので，このためのソフトウェアがバッチ・システムと異なるのはいうまでもないことであり，グラフィック・システムではソフトウェアが非常に重要な役割を果たしている．とはいえ，グラフィ

ック・システムのソフトウェアについて一般的に論ずることはかなりむずかしい．それは現在のグラフィック・システムにおいては，まだ一般的に受け入れられるソフトウェア・システムの概念が，必ずしも固定されていない状態にあるからである．特殊な分野に目的をしぼった実用システムは存在するが，これはほかの分野の問題には利用しにくい．グラフィック・システムにはたして普遍的なソフトウェア・システムというものが存在しうるかどうかすらが問題にされているのである．しかし以下では問題を多少抽象化し，ある程度幅広い問題に対応しうるソフトウェア・システムとして考えられる形式について論じる．

グラフィック・システムは，その本来の性格から，利用者が問題を解くために，その都度プログラムする必要なしに，すでにシステム内に準備されている諸プログラムを利用するのみで十分であるようなものが理想的な姿とされる．さらにシステムのオペレーションの方法を，必要に応じてプログラムが教えてくれるものであることがより望ましい．すなわち，ユーザは計算機に関しては素人であり，プログラム開発は別の専門のプログラマに任せられる．このような状態を想定したときに必要になるプログラムの種類は数多いが，それらは大体三つのレベルに分けられる．第一はシステム全体を管理するオペレーティング・システムであり，第二は図形処理用の共通プログラムであるグラフィック・システム・ソフトウェアであり，第三は個々の問題ごとのアプリケーション・プログラムである．

オペレーティング・システムはシステム内のすべてのプログラムのスケジュリング，その実行の管理，リソースの割当てなどを行なう．この基本的な概念は従来の OS と特に変わりがない．ただし，現在の TSS やマルチ・プログラミング・システムがこのままでグラフィック・ターミナルを含むシステムに対しても十分であるというわけではなく，リソースシェアの方法やそれに関連したジョブの管理などの面で改善の余地がある．

グラフィック・システム・ソフトウェアは，グラフィック・システムの最も特徴的といえる部分である．これはデータ構造に関するもの，図形の生成に関するもの，割込みの処理に関するものなど多くのルーチンから成り，アプリケーション・プログラムにより利用される．

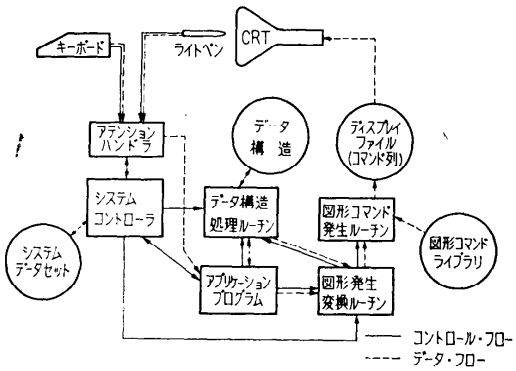
アプリケーション・プログラムが参照し，あるいは操作を施すデータは構造化されて記憶装置におかれる．これをデータ構造と呼ぶ．データ構造はユーザの問題の計算機内モデルであるが，与えられた種々のデータを集積してデータ構造を作り上げたり，データ構造をたどって特定のデータを取り出したり，データ構造の一部を削除したり，修正したりあるいは挿入したりするルーチン群が準備される．これらのルーチンはデータ構造の形式に依存することはいうまでもない．

データ構造はユーザとアプリケーション・プログラムとの間のデータの仲介役を果たしており，ユーザの提示する問題はデータ構造を介してアプリケーション・プログラムに送られ，計算が終了するとアプリケーション・プログラムは結果をデータ構造に書き込む．それを表示するためには，表示用のいくつかのルーチンの助けを必要とする．

通常，個々の図形データがデータ構造内に記憶されている形式と，それがディスプレイ装置に与えられる形式とではかなり異なっているので，データを図形で表示するにはその変換が必要である．そのために図形発生用のルーチンが準備される．また自動車や船といった物体の形を表現するデータの場合，データ構造内では座標値のセットが記憶されているが，それを画面に図形として表示する際に，図形を縮尺（拡大）して見るとか，移動や回転して見るとか，あるいは透視図として見るといった様々な要求が生じ，そのために表示用のディスプレイ・ファイルを作成する以前にデータ間の様々な変換が必要になる．図形表示ルーチンにはこれらの変換ルーチンも含まれる．

グラフィック・システムではオペレータの制御のもとで仕事が進められ，このオペレータの意向は，入力装置の割込み機能を介して計算機に伝えられる．計算機内では割込みは入出力制御ルーチンを介してアテンション処理ルーチンに送られ，ここで分類整理され，優先順位に応じてサービス・ルーチンを呼び出す．

アプリケーション・プログラムはユーザの問題向きのプログラムであり，システムの利用目的の範囲内で多種類のものが準備されていることが必要である．アプリケーション・プログラムの中心部分，すなわち問題に関する各種の演算や変換を行なって実際に問題を解く部分は，バッチ処理におけるユーザのプログラムと同様であるが，これと同時にインタラクションの機能を有し，その実行はオペレータの制御のもとにおかれ，パラメータなどは必要に応じてオペレータから供給される形式にプログラムされている必要がある．これにはアプリケーション・プログラムの方が入力を要



第12図

求する場合も，オペレータの方が割込み機能を用いて実行中のプログラムを中断し，それまでと異なった動作をさせたり，パラメータを変更したりすることもある．このようにアプリケーション・プログラムとオペレータとの間には，コミュニケーションの方法が確立されていなければならない．

以上のソフトウェアの全体の構成を図示したのが**第12図**である．ソフトウェア・システムを構成している各要素間は，矢印で示されたようにデータおよび制御の両面で相互に関連し合っている．

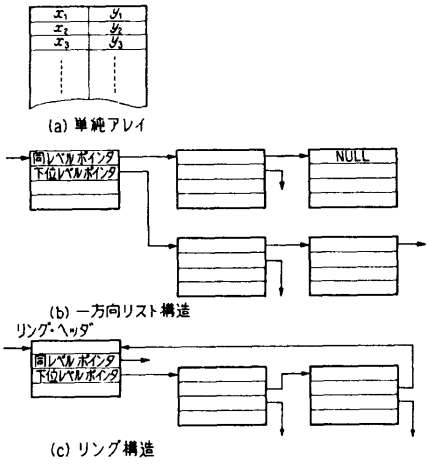
ここで示されたのは，一つのアプリケーションが実行されているときに，システムの有すべき機能とそれを遂行するプログラムの関係を示したものであって，必ずしもどのアプリケーションにも共通して用いられ，したがって，システム・ソフトウェアとして含まれるべきグラフィック・システム・プログラムが存在することを意味するものではない．たとえば，データ構造についてみると，汎用のデータ構造を定義してその処理ルーチンをシステム・ソフトウェアに含んでいるものもあるが，現在のところではデータ構造とその処理ルーチンは，アプリケーション・プログラマの責任において，アプリケーション・プログラムと同時に作成される場合が最も多い．広い範囲の問題に共通して利用できる効率のよい汎用データ構造が見い出されれば，データ構造処理をシステム内に含めることによりアプリケーション・プログラムの作成が容易になるが，一般論としてデータ構造の汎用性と効率は相反する性質のものであるため，汎用のシステムでは問題処理の効率が低下し，実用面からみると一般化への方向は必ずしも成功していない．データ構造の一般化か個別化かへの問題は，アプリケーション用の言語の問題ともからんでくるので，新しいデータ構造を見い出だすという可能性を含めて，今後も検討が必要である．ここでは以上の問題をもう少し詳細に述べておこう．

3.4.1 データ構造[30),39),84)]

グラフィック・システムでも，人と計算機とのインタラクションが少なく，問題もそれほど複雑でない場合には，データ構造も単純なもの，たとえばテーブルとか二次元配列といったもので十分間に合うであろう．CRT 画面上の図形は図形コマンドの列としてバッファにたくわえられている．この図形の各要素はユーザによって名前がつけられ，その名前とその名前によって表わされる図形要素の実体，すなわちバッファ上のアドレスの間の関係がこの配列にたくわえられる．これはコリレーション・リストと呼ばれるがライトペンが用いられたとき，このコリレーション・リストを参照することにより，検出された図形要素の名前を知ることができる．

しかし，もっと複雑な問題，たとえば図形の挿入・削除・修正などが行なわれたり，サブピクチュアを定義したり，それを変換するといった応用では，あるリスト内の一要素がそれ自身ほかのリストとなっているといった，もっと複雑な階層をなすデータ構造が必要になってくる[75)]．ここでサブピクチュアとは全体の図形の一部分であるが，それ自身がいくつかの基本図形要素からなるまとまった一つの意味のある図形のことである．たとえば，回路図におけるダイオードの記号などがこれに相当し，これ自身ベクトルなどいくつかの基本図形要素から構成されているが，回路図の中に繰り返し出現し，物理的にも意味のある単位を表わしている．このような複雑な構造が必要となるのは，扱う問題を表わす多数のデータが相互に関連を有しているからで，上述の回路図の表現でも，トランジスタ，ダイオード，抵抗，キャパシタなどが相互に接続されることにより特定の回路を構成し，これら回路要素の特性値を示す諸パラメータも互いに関連している．

この種の構造では，各要素ごとに固有のデータを保有するデータ・ブロックというものが作られ，各要素間の関連はポインタにより示されるのが普通である．ポインタはデータ・ブロック内におかれていて，このブロックと関連づけられる次のブロックのアドレスを示す．ポインタをたどって関連する要素を読み出すことができ，またポインタを変えることにより，データ構造を作り変えることも比較的容易である．この際配列と異なってブロックは記憶装置内でどこにでもおか



第 13 図

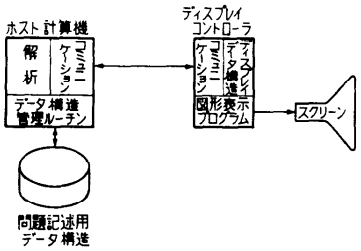
れるから（ポインタの表示しうる範囲内で），構造の一部を変更してもほかの部分に影響することが少ない．しかしこのような方式では構造が複雑になるにつれてポインタの数が増し，そのために大量の記憶容量を必要とするという欠点がある．

第 13 図に代表的なデータ構造の例をいくつか示す．矢印はポインタを示す．ポインタを有するものでも一方向にのみ関連がつけられ，NULL シンボルで終わる LISP 型のリストや，両方向ポインタを有するもの，あるいは同レベルの関連の最後のポインタは上位ブロックにもどるリング構造などがある．リスト構造は言語処理などによく用いられるが，図形処理では下位から上位へと関連をたどることも多いので，一方向リスト構造では不十分である．またすべて両方向のポインタを有するものは，オーバヘッドが大きくなる．そこで通常，図形処理で用いられるものにはリング構造が多い．リング構造の処理言語としてこれまでにも SKETCHPAD[75], CORAL[77], APL[23], ASP[49], MASP[39] など各種のものが発表されている．

ところでこれまでのデータ構造の考え方は，図形表示用のデータと問題の記述用データの両方を同一のデータ構造で表現しようとする傾向が強かった．電気回路図などを例にとれば明らかなように，図形というのは人間同志の一つの約束に基づく問題の表現法である．この表現はこの問題を解析するもの，すなわちアプリケーション・プログラムにとっては無意味であって，問題解析のためには別の形式の問題表現が必要となる．このように，データ構造は人間に理解できる形式とプログラムに理解できる形式の両方をかねそえた，いわば問題表現の変換部分である．最近この二つの面を一つのデータ構造で受け持たせるのは，構造を複雑化し，データ・ブロックとして一般的には可変長のものが必要となる傾向が強いという意見があり，この二つの問題表現を分離して二つの別個のデータ構造を持つものが現われてきた（GPLI, LEAP[68] など）．これらのシステムではディスプレイ用のイメージ情報に単純なツリー構造を用い，問題の表現用には別のデータ構造を備えるといったように，データ構造の有する機能の二面性を明確に分離している．この両構造は密接に結合されていることはいうまでもない．このようなダブル・データ構造の具体化に際しては，構造内の要素を見い出すための識別情報はデータ構造の機構に依存しないシンボル形式が用いられる必要がある．こうしておけばどちらかが片方の構造を作りかえても（たとえばキャベジコレクションにより）相手の構造まで変更する必要がない．システム構成のところで述べたようなディスプレイ・プロセッサを有するシステムでは，表示用データ構造はディスプレイ・プロセッサに，問題用データ構造はホスト計算機におかれるという可能性があり，この際に上記の条件が特に重要になる（**第 14 図**）．

データ構造については，さらに補助メモリを利用する場合のことも考慮しておく必要がある．ドラムやディスクなど補助メモリを利用する場合には，データ構造の利用効率が著しく低下するので，実用面からは最も重要な問題の一つとなる[78]．

補助メモリを利用する際にとられる手法の一つは，バーチュアル・メモリの概念の導入である．すなわちユーザからみたときには主メモリも補助メモリも区別なく，連続したバーチュアル・メモリとして扱われる．物理的にはこれはページ単位で自動スワッピングがなされ，アソシアティブ・レジスタの利用やページ先取りなどの方法を用いて，この利用効率を増すことができる．この方法は特にデータ構造のためのものでな



第 14 図

く，これ以前にすでに TSS において用いられている方法である．

もう一つの方法はハッシュ構造と呼ばれるものである．これはたとえばディスクのようなメモリに記憶されている要素を捜しだすのに，リスト・サーチの方法によると非常にむだが大きいため，これを省き，情報取り出しを高速で行なう方法である．ハッシングとは大きな範囲内のとびとびの番号を，できる限り衝突確率の小さくなるように小さな範囲に変換する方法であり，これは在庫管理などの応用ではすでに普通の方法として用いられている．これをデータ構造に取り入れるには，まず概念の表現を，“ある対象(O-Object)の属性(A-Attribute)はある値(V-Value)である”という基本形式に表わすということが根底にある．(A(O)=V)．たとえば JIRO(O) の ANI(A) は TARO(V) であるといった関係である．そこでファイル・スペースを二次元配列のように考え，配列の内容には値 (V) を含み，行および列はそれぞれ属性 (A) および対象 (O) を表わすとする．A とか O は通常広い範囲にとびとびにしか存在しないので，ハッシュ変換のオペレータはこれら二つを合わせて，もっと小さい一次元の範囲 (ファイルの容量内) に変換する (H(A, O)＝ファイル・アドレス)．A(O) の値 (V) が多数存在するときには配列のセル内には値そのものでなく，要素のリストへのポインタを入れる．ハッシングはこのように広い領域から狭い領域への (中への) 変換であるから，異なった (A, O) でも同一のセルをさす場合がある．この場合にも同じセルを示した関係をポインタで結んだリストを作る．また記憶方法に冗長性をもたせることにより，A(O)＝V の関係で V 以外の要素を見い出すという要求にたいしても，アクセスの回数を減少する方法が用いられている．

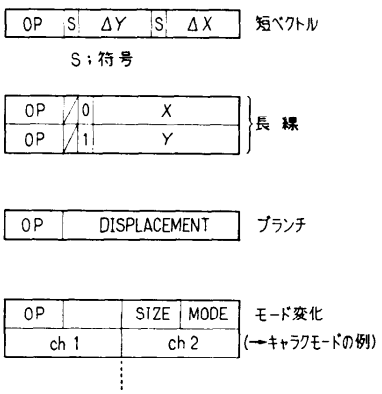
以上述べてきたことからも理解されることと思うが，データ構造の問題はグラフィック・システムのように，問題を直接に提示することができ，かつこれをインタラクティブに解くことを目的としたシステムのソフトウェア開発に最も大きな影響を与える中心的な問題である．それにもかかわらず，というよりむしろそのために，これまで決定的といえる方式は現われていない．現在は，まだ一般的なデータ構造を使うのがよいか，アプリケーションごとのものがよいかということすら，十分な解答が与えられていない．しかし，これまでの経験に基づいた次の三つの主張は十分留意に値する．すなわち，(1)システムはユーザがそれぞれのアプリケーションに適した構造を用いられるような言語を有すること，(2)一般的なデータ構造が作られるとしたら，補助メモリ上でのリストまたはリング構造を用いたものであること，(3)ハッシュ・コード法を用いた連想記憶方式は，一般的なデータ構造に対する最良の方法であること．このような主張に基づいて，よりよいデータ構造およびその処理ルーチンを見い出すには，今後の研究にまつところが大きい．

3.4.2 図形発生および変換

3.4.1 で述べたように，データ構造は与えられた問題に関するすべてのデータを構造化してたくわえている．そして解析のための問題の表現という，アプリケーション・プログラム向きの面と，図形表示のための問題の表現という人間向きの面を兼ね備えている．アプリケーション・プログラムは問題の計算機内モデルであるデータ構造から必要なデータを受け取り，解析を行なった後計算結果を返してくる．これらの情報は次いでディスプレイ面に表示されなければならないが，データ構造から必要な情報を取り出しても，それを表示するには一般にいろいろの変換が必要であり，そのためのルーチンが準備される．アプリケーション・プログラムはこれら図形発生や変換のルーチンをコールし，これから作られる図形情報がディスプレイ・バッファに送られる．このような変換ルーチンが必要になるのは，データ構造内にはデータはできるだけ冗長度が少なく，効果的にたくわえられており，これがそのまま図形表示情報とはなり得ないこと，データ構造は数学的に定義づけられた座標系に基づいており，これがディスプレイ面の座標系とは一致しないなどのためである．これについてもう少し詳しく述べておこう．

(i) ディスプレイ・ファイルの作成

すべての図形は基本となる単純な図形要素に分解され，この各図形要素は一つの図形コマンドで表わされる．したがって，計算機内では図形とは図形要素を表わす図形コマンドの集まりであり，本稿ではこれをディスプレイ・ファイルと呼んできた．ディスプレイ・ファイルはディスプレイ・コントローラが直接アクセスできる記憶装置内におかれる．各図形コマンドはディスプレイ面に固定された座標系に基づいて図形要素を表わしている．つまり現用の CRT ターミナルではディスプレイ面は通常 XY 両方向に 512～4096 の格子点を指定できるようになっているが，たとえば左下すみを (0, 0) すなわち原点とし，右方に X 軸，上方に Y 軸をとったものがディスプレイ面の座標系になって

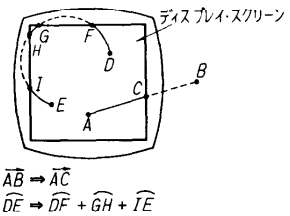


第15図

いる．図形コマンドにはデータとして X, Y 両座標値を含むが，この値はディスプレイ面上の座標系上の位置である．図形コマンドの一例を**第15図**に示す．この例で，たとえば長線というのは現在のビーム・ポジションからコマンド内の X, Y データで示される位置まで直線を引くという指令であり，ベクトルは現在のビーム位置から ΔX, ΔY だけ変化した位置までベクトルを引くことを指示している．通常このような座標データのほかに，輝度の指定や，線の種類（実線・破線・点線など）そのほかの制御情報を含み，またジャンプ命令，サブルーチン・リンク命令などフローの制御に関するコマンドも含まれる．データ構造内からの図形情報に基づいてディスプレイ・ファイルを作成するルーチンがディスプレイ・ファイル作成ルーチンである．データ構造内では，たとえば円を表現するためには中心座標と半径をデータとして含んでいれば十分であるが，これを表示するには円を小ベクトルの列として表現しなければならない．ディスプレイ・ファイル作成ルーチンはデータ構造からのデータとして，“円である”という情報と，それを定義する2個のパラメータから，小ベクトルの図形コマンド列を作り出す．このほかディスプレイ・ファイルに関しては各種の管理ルーチンが必要である．たとえば画面上での図形の部分消去などによってディスプレイ・ファイルは虫喰い状態になるのでギャベジコレクションを行なったり，新しく図形要素が加えられるとき，あいている場所を割り当てるなどの仕事が行なわれる．

（ii） 図形変換

データ構造からのデータがディスプレイ・ファイルの作成に用いられるまでには一般には途中で変換処理を受ける．すでに述べたように，データ構造の座標系とディスプレイの座標系とは必ずしも一致していない．たとえば船とか建造物の設計に際して，図面をデータ構造として計算機内に記憶すると仮定してみよう．この場合データ構造の用いる座標系は設計される物体に固有のものとして定義される．一方これを表示する際に，ディスプレイ面の大きさには限度があるので，一部分を拡大して表示したり，表示する部分を平行に移動したり，視点をかえたときの形をみるために回転したりといった操作が必要になる．この様子はズームの可能な，視野の狭い窓を通して，向こう側の大きな図面をのぞき見るようなものである．この意味でディスプレイの表示面を“窓”と呼んだりする．いずれにしろ，このような変換が行なわれるときにはディスプレイ面座標は各変換操作ごとに異なってくる．窓の平行移動はデータ構造座標系に対するディスプレイ座標系の原点の移動を意味し，拡大・縮小は座標のスケールの変更を意味し，回転は座標の回転を意味する．したがって，データ構造内のデータはこれら変換に対応して変換処理をうけ，ディスプレイ座標値に変換されなければならない．これらの変換はマトリックス演算により行なわれる[1]．さらに，実際に表示する前にどの図形要素がディスプレイ面の表示範囲にはいるかを調べておく必要がある．この範囲に全然含まれないものもあり，部分的に含まれるものもある．部分的に含まれるものについては，画面内にはいる部分だけ図形コマンドが作られるように座標値を変換しておかなければならない．たとえば直線 $\overline{AB}$ のうち $\overline{AC}$ のみ（**第16図**）画面にはいるときはデータ構造内のデータは $\overline{AB}$ であっても，$\overline{AC}$ というデータを作ってからディスプレイ・ファイル作成ルーチンに送ってやる．これはシザリングと呼ばれる操作であるが，曲線の場合などでは画面の境界との交点を見い出すことがかなり複雑になり，大きな図面に相当するデータを含む場合この処理だけでも大きな時間を費やしてしまう．そこで上述のような座標変換やシザリングを専用のハード



第16図

で高速に処理しようとする試みがある．前者についてウェアはマトリックス・マルチプライヤとかベクトラン[40]など，後者についてはクリッピングディバイダ[73]といった装置が開発されている．

なお，図形処理で扱われる図形としては，直線とか単純な二次曲線といったものが最も多い．これらをデータ構造内にたくわえるには直線としてはたかだか両端の座標値に相当する2組の座標値，二次曲線でも3組の座標値があれば十分である．しかし，かりに一般の曲線の場合には，このような簡単なパラメータ化した表現がとれないので，サンプル値の組をそのまま記憶するということもある．この際はサンプル値から元のなめらかな曲線を作り出す関数発生ルーチンなども必要となる．さらにこれに関連して，図形入力データから最良のサンプル値を取り出すルーチンとか，さらに一般曲線を一定の誤差内でパラメータ表現のできる曲線で近似させるルーチンなども必要となるであろう[41),42)]．

3.4.3 高レベル言語[48),53)]

グラフィック・システムにおいて，高レベル言語といわれるものにはいろいろのタイプのものがある．それらを分類してみると，まず問題向きかあるいはプロセジュア記述型かの2種類があり，さらにその表現の形式として，従来どおりの文字列でステートメントを表わすもの（記述式）と，ファンクション・キーやライトペンの操作でステートメントを指定するもの（指示式）とがある．これまで述べてきたグラフィック・システムのイメージは，ユーザ側からみて理想的な形態として問題向きの指示式言語の存在を暗に想定していたといえる．実用上からは，たとえ問題向き言語であっても指示式のみに限定されるのは不便なこともあり，両方の形式が利用できることが望ましい．指示式と記述式とでは利用面では異なるが，実際には同一言語の二つの形式であり，この間に1対1の対応をつけることができる．したがって，言語表現の基本としては記述式を考えておけば十分であろう．

また問題向きの言語の作成にはアセンブラを用いてもよいし，現実にもアセンブラが使われる例が多いが，このためにプロセジュア型の言語を利用することもできる．

この意味で上述の4種のタイプのうちプロセジュア型記述式言語は最も基本的なものであり，他の言語はいずれも仕様がきまればこの言語を用いて作成することができる．

このようなグラフィック・システムのプロセジュア型言語は問題の処理手続きを記述する従来の言語機能のほかにインタラクティブな使用を可能にするためのいくつかの機能を備えていなければならない．すなわち，(1)アテンション処理機能，(2)図形表示用データ構造処理機能，(3)問題記述用データ構造処理機能，(4)図形生成・変換機能，(5)計算機間コミュニケーション機能である．

これらはいずれも従来の言語には含まれていないので，従来の言語を新しい機能を追加するか，あるいは新しい言語を開発する必要がある．前者にはたとえばFORTRAN にインタラクションの機能をサブルーチンとして付加したFORTRANグラフィック・サブルーチン・パッケージ(GSP)[69]のようなものがある．これは従来のFORTRANに必要最小限の機能を付加したもので，グラフィック・システム用の高レベル言語と呼べる範囲で最も単純なものである．後者の例としては，たとえばD. T. Rossらによって開発されたAEDがある．これは一般的なProblem Solverとしての機能をも含む最も高度な言語の一つである[64)~67)]．

コンソールからのファンクション・キーやライトペンなどの割り込みに対するアテンション処理はインタラクティブな利用を可能にするためアプリケーション・プログラムが有すべき最も基本的な機能である．アテンション信号が生じたとき，システムはアテンション処理に必要な最小限の情報を集めて，ユーザのアテンション処理ルーチンが利用できるようにしておくことが必要である．アテンション信号に伴う情報としてはデバイス番号，アテンションのタイプ，キー番号，ライトペンの検出した図形名またはコマンドアドレス，座標値などがある．このようなアテンション割り込みに対してプログラムがとるべき動作の定義づけをするにはいろいろの方法があり，アテンションが生じたときのプロセジュアを定義する方法と，アテンションの発生によりこれらのプロセジュアが呼び出される方法に相違がある．アテンションの取扱いには，(1)割込みが生じたら直ちにアテンション処理にはいるもの（非同期処理）と，(2)プログラムのテスト形式でアテンションの有無を調べる（ポーリングを行なう）もの（同期処理）がある．後者の方式ではアテンションは実質的にはプログラムによるテストに転化されるから，従来のIFステートメントを用いたり，またサブルーチン・コール形式でアテンションのタイプとその処理ルーチンとの対応をあらかじめ定義しておき，

表にしておく方法もある．一方，PL/1 でアテンションの処理をオーバフローや 0（ゼロ）ディバイドの処理と同じように，ON ステートメントで処理するように拡張する方法もあり，またシステムの状態とアテンションによる状態遷移という概念を用いて，各状態ごとに複数種のアテンション・タイプの生起を許すようにしたものなどがある．

アテンション以外の機能に関しても状況は大体似たようなもので，システムが備えている機能とアプリケーション・プログラムを結合するための言語機能が付加される．このようなシステムの設計において重要なことは，アプリケーション・プログラマがシステム内部に精通していなくても十分プログラムできることである．たとえば，図形表示用データ構造処理機能についてみると，プログラマは表示される図形要素の種別（直線，円弧など）とそれを構成するのに必要なデータ（直線の場合2端点の座標，あるいはこれに相当する情報を与えるすでに定義された要素名など），その要素に与える名前など外部情報はすべて定義してやる必要はあるが，それが内部ではどのように構造化されているかについて知らなくてもよいようなプログラム・システムであることが必要である．

3.5 コンピュータ・グラフィックスの応用と現状

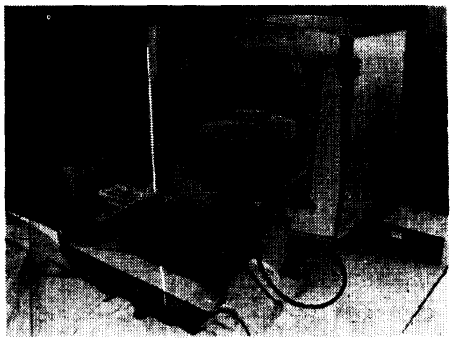
最後にコンピュータ・グラフィックスの応用面からみた現状を簡単にしるしてこの稿を終了したい．

コンピュータ・グラフィックの応用分野は，従来の計算機利用の分野をそっくり含む．もっと広範囲なものになっている．これらをいくつかに分類して簡単に示す[35]．

（i） 設計への計算機利用 (CAD–Computer Aided Design)[38),44),58)]

コンピュータ・グラフィックスに最も大きな期待が寄せられている応用の一つが CAD である．設計というプロセスに計算機の力を活用すること，さらに設計から生産いたるまでを一貫化することは設計合理化の目標であるが，これはコンピュータ・グラフィックスの技術によって初めて可能になる．設計問題の代表的な例は IC マスクの設計，船・自動車・航空機などの形状設計，道路・ダム・橋梁，あるいはビルの設計，配線や配管設計など静的な形状の設計に関するもの，カムやリンクなどの機構設計のように，動的な問題として扱われる種類の応用がある．

形状設計においては三次元物体の形状処理がむずかしい．物体の画面上の表現法としては三面図，斜影図



第 17 図

あるいは透視図などの方法があり，この間の変換には特に困難はない．問題なのはこのような三次元形状を記述し，かつ記憶する形式や，自由曲面の生成技術[1),18),37)]，物体の Hidden Line 処理技術や中間色表示技術[2),3),8),63),86)]などである．空間内の座標値やその点の勾配など，与えられた条件を満たすなめらかな曲面の生成と表示には，従来の曲面論と全く異なる自由曲面理論を必要とする．これには S. A. Coons[18)] の導びいた Coons 式が広く用いられており，また，これと独立に Coons 式を含むより一般的な表現を穂坂[37)]が導びいた．Hidden Line は物体の後方にかくれた部分を判別して，画面上から消去する技術である．これはパターン識別能力のない計算機には時間のかかる面倒な問題であり，これまでにもいろいろの方法が発表されているが，いずれもかなりの計算時間を必要とする．

設計分野でも比較的設計初期の研究開発に属する問題にもグラフィック・システムは重要な手段となりつつある．この段階ではシミュレーションが重要な手法である．上述の機構設計もこの一種であるが，このほか回路設計[9),21),22)]，構造解析，振動解析，翼の空力特性，航空機の着陸時のシミュレーションなどが行なわれる．さらにシミュレーションは経済問題とか道路交通管制，飛行場運行管理などの管理問題その他，かなり広く用いられるコンピュータ・グラフィックスの重要な手法である．

一方，設計から生産への移行時の問題としては，部品設計と NC（数値制御）機械用入力テープの自動作成が行なわれている．

このような設計プロセス合理化の最終目標は，設計から生産までの一貫化にあるが，これを押し進めてゆくと，単にインタラクティブに図形を扱うというだけでなく，標準部品，設計規格，関連法規，各種設計デ

ータなどをデータ・ベースとする情報検策システムや，資材在庫，工程の諸管理を行なうシステムと一体化された大形のコンピュータ・システムの開発が必要とされることになる．現在まで，このような大形システムを具体化した例はみあたらないが，これを構成する諸技術は個別的にはすでに数多く実現している．

(ii) 指令・管理システム

グラフィック・システムの第一の特徴は，図形によらなければならない設計の問題を扱えることであるが，多量の情報の表示が可能であることは，人間の意志決定の手段として非常に有用であることを意味している．したがって，指令・管理システムとしてグラフィック・システムが適していることは明らかである．実際，図形処理のはしりが防衛用指令管理システムであるSAGE システムにみられることはすでに述べたとおりである．

SAGEのような軍用のシステムのほかにも，アメリカ NASA では宇宙開発計画に全面的にグラフィック・システムを利用しているし，日本では国鉄が鉄道の運行管理に用いようとしている．道路交通管制などの利用も考えられている．

グラフィック・システムの利用の対象として，設計問題は第一に想起される問題であるが，現実には設計には単に図形を表現するだけですまない多くの問題があり，実用化の速度が必ずしも順調とはいえない．これに対し指令管理システムは，問題の範囲がかなり明確化されることもあって，設計の場合ほど技術的な困難がない．したがって，今後も大きなシステムの運行管理などに利用されることと思われる．

(iii) 経営情報システム

経営問題も人間の意志決定の問題として捕えるなら，グラフィック・システムの有用性は指令管理システムと同様である．経営情報システムとしてディスプレイを用いるときの使い方は，たとえば，経営者が製品や企業活動に関する情報を要求すると，それがディスプレイに表示される．経営者はそれを見てその中から特定の項目を拾い出し，そのより詳細な資料を要求するというぐあいに進められる．このような使い方では，計算機内には高度に構造化された形でデータ・ベースが整えられている必要があり，質問を理解し，返答を作成するプログラムが利用できるようになっていなくてはならない．

(iv) オンライン情報システム

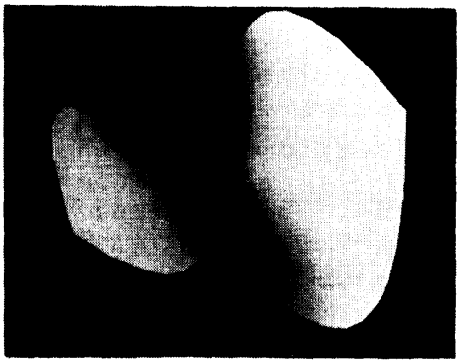
ここでオンライン情報システムというのは医療情報システム，予約システム，オンライン・バンキング・システムなどであり，元来はキャラクタ・ディスプレイでことたりている分野であるが，医療情報システムなどでは患者の状態を示す情報を図で表示することにより，病状に対する医者の判断をより一層正しいものにする．

以上のほかにグラフィック・システムはプロセス制御のモニタ，新聞などの編集，教育用 (CAI)[80]，情報検索[16]そのほか数多くの分野での応用の可能性を有している．

最後にコンピュータ・グラフィックス技術の現状をごく簡単に述べておこう[32]．現在，この分野で技術的にも，実用化の割合でも最も進んでいるのがアメリカであり，日本とヨーロッパ諸国がこれを追うという一般的な図式が成り立っている．アメリカにおいても技術レベルは職種によりまちまちで，自動車産業が先端をゆき，航空機・電機・造船・建築などがこれを追っている．特にGM社においては，大規模な設計プロセスの合理化計画が進行中でその成果が注目される．しかし，一般的にはグラフィック・システムはまだコスト高であり，産業界ではすでに開発された技術をいかに活用して，これをペイする段階に持ってゆくかに主力が注がれている．しかも現状では，NC 関係など一部を除くとインタラクティブなディスプレイが生産部門に組み入れられるまでには至っておらず，リサーチ部門での利用がおもになっている．生産部門では自動製図機やディジタイザなど，図面の処理の合理化が先行しているのが実情である．このような傾向は日本でもほぼ同様である．

一方，大学や研究所など研究機関においては，図形処理の新技術の研究が進められ，三次元物体の表示やその処理技術が意欲的に進められている．物体にHidden Line 処理を施し，影をつけ，Perspective に表示したり，影つき曲面体を表示したり（**第18図**：アメリカ，ユタ大学）あるいはそれらをカラーで表示するなどの実験がなされている．これらの技術は原理的には完成しているといえる．このほか基礎的な面では，データ構造や言語の研究が進められる一方，アプリケーション面でも三次元構造解析が物体の展開処理，写真解析や IC マスク設計，建築デザインなど，実用に近い各種の研究がなされている．

ハードウェア面でもマトリックス変換やシザリングを行なう専用装置が開発され，ストレージ管の利用を含めた低コスト端末についても関心が高い．



第18図

4. むすび

以上コンピュータ・グラフィックスに関して要点のみ簡単に述べてきた．実際にはここで問題にした事柄の多くは，グラフィック・システム固有の問題というよりも，対話型処理システムに共通の問題として生ずるものである．しかし，グラフィック・システムは代表的な対話システムであり，しかも，人間との情報交換速度が著しく速い．このことはこれら諸問題をグラフィック・システムにおいて特に顕著にしている．

これらの問題の多くはまだ十分に解決されていない．データ構造ひとつとってもまだグラフィック・システムのあるべき姿についての考え方は定まっていない．このため，元来システムで備えているべき多くの機能を，現状ではアプリケーション・プログラマが開発せざるを得ず，これがシステムの実質コストを増大する原因にもなっている．

グラフィック・システムのソフトウェアの開発速度が遅いということには，ディスプレイ・ターミナルの普及度がまた低いことにも大きな原因がある．これはディスプレイ・ターミナルのコスト高によるのであるが，量産によりまだコストの低下する可能性はあるわけで，それにはコストエフェクティブネスを改善し需要の増大をまたなければならない．そのためにも，システム，データ構造，言語などの面で今後も多くの問題を解決すると同時に，役に立つアプリケーション・プログラムを開発してゆくことが必要である．

コンピュータ・グラフィックスのように，広い分野に関連する技術をまとめて解説することはむずかしいことであり，ごく表面的な事柄に触れるのみで，すでに予定枚数をはるかにこえてしまった．しかも，各項目の内容は中途半端なものになってしまったことを最後にお詫びする次第である．

参考文献

1) Ahuja, D. V. & Coons, S. A., "Geometry for construction and display", IBM Systems Journal, Vol. 7, No. 3 & 4, 1968.
2) Appel, A., "The notion of quantitative invisibility and the machine rendering of solids", Proc. A. C. M. National Meeting, 1967.
3) Appel, A., "Some techniques for shading machine renderings of solids", Proc. SJCC, 1968.
4) Appel, A., Dankowski, J. P. & Dougherly, R. L., "Aspecy of display technology", IBM Systems Journal, Vol. 7, No. 3 & 4, 1968.
5) Ball, N. A., Foster, H. Q., Long, W. H., Sutherland, I. E. & Wigington, R. L., "A shared memory computer display system", IEEE Transactions on Electronic Computers, Vol. EC-15, No. 5, Oct. 1966.
6) Beckermeyer, R. L., "Interactive graphic console environment and software", Proc. FJCC, 1970.
7) Blatt, H., "Conic display generation using multiplying digital-analog decorders", Proc. FJCC, 1967.
8) Bouknight, J. & Kelley, K., "An algorithm for producing half-tone computer graphics presentations with shadows and movable light sources", Proc. SJCC, 1970.
9) Breuer, A., "General survey of design automation of digital computers", Proc. of the IEEE, Vol. 54, No. 12, Dec. 1966.
10) Chasen, S. H., "The introduction of man-computer graphics into aerospace industry", Proc. FJCC, 1965.
11) Chasen, S. H. & Seitz, R. N., "On-line systems and man-computer graphics", Computers and Automation, Nov., 1967.
12) Chin, F. C. & Dougherty, R. L., "A system for implementing interactive applications", IBM Systems Journal, Vol. 7, No. 3 & 4, 1968.
13) Chesler- L. & Turn, R., "The application of on-line graphical techniques for programming and operating a "moving network" moniter-ing display", The RAND Corp., Memo. RM-5183-PR, Jan. 1967.
14) Childs, D. L., "Description of a set-theoretic data structure", Proc. FJCC, 1968.
15) Christensen, C. & Pinson, E. N., "Multi-function graphics for a large computer system", Proc. FJCC, 1967.

16) Coles, L. S., "An on-line question-answering systems with natural language and pictorial input", Proc. 1968 A. C. M. Natinal Conference.
17) Coons, S. A., "An outline of the requirements for a computer aided design system", Proc. SJCC, 1963.
18) Coons, S. A., "Surfacesfor Computer-aided design of space forms", MAC-TR-41, June 1967.
19) Corbin, H. S., "A survey of CRT display consoles", Control Engineering, Vol. 12, Dec. 1965.
20) Davis, M. R. & Ellis, T. D., "The RAND tablet: A man-machine graphical communication device", Proc. FJCC, 1964.
21) Dawson, D. F., Kuo, F. F. & Magnuson, W. G., "Computer-aided design of electronic circuits: A user's viewpoint", Proc. the IEEE, Vol. 55, No. 11, Nov. 1967.
22) Dertouzos, M. L. & Therrien, C. W., "Circal: On-line analysis of electronic networks", Report ESL-R-248, Oct. 1965.
23) Dodd, G. G., Beach, R. C. & Rosssl, L., "APL-Associative programming language", Research Publication GMR 622 July 15, 1967.
24) Faiman, M. & Nievergelt, J., "Pertinent concept in computer graphics", Univ. of Illinois Press, 1969.
25) Feldman, J. A., "Aspect of associative processing", MIT Lincoln Laboratory, TN 1965-13, Apr. 1965.
26) Fisk, C. J., Caskey, D. L. & West, L. E., "ACCEL: Automated circuit card etching layout", Proc. the IEEE, Vol. 55, No. 7, Nov. 1967.
27) Frank, W. L., "Software for terminal-oriented systems", Datamation, June 1968.
28) Gallenson, L., "A graphic tablet display console for use under time-sharing", Proc. FJCC, 1967.
29) Gellert, G. O., "Geometric computing: Electronic geometry for semiautomated design", Machine Design, Mar.-Apr. 1965.
30) Gray, J. C., "Compound data structure for computer aided design; a survey", Proc. A. C. M. National Meeting, 1967.
31) Green, R. E., "Computer graphics", Computer Aided Design, Spring 1970.
32) Green, R. E., "Computer graphics; a status report from the USA", Computer Aided Design, Summer 1970.
33) Gruenberger, F., "Computer graphics", Thompson Book Company, 1967.
34) Hlady, A. M., "A touch sensitive X-Y position encoder for computer input", Proc. FJCC, 1969.
35) Hobbs, L. C., "Display applications and technology", Proc. IEEE, Vol. 54, No. 12, Dec. 1966.
36) 穂坂 衛: "コンピュータ・グラフィック・ディスプレイ", 電子通信学会誌 51, 5, 1968.
37) 穂坂 衛: "曲線, 曲面の合成および平滑化理論", 情報処理学会 Vol. 10, No. 3, 1969.
38) 穂坂 衛: "コンピュータ・グラフィックスの機械工学への応用", 電子通信学会誌 53, 4, 1970.
39) 穂坂 衛: "データー構造", 電子通信学会誌, 53, 4, 1970.
40) 穂坂 衛, 柿下尚武, 槻木公一: "三次元動的表示とベクトル演算装置について", 東京大学宇宙航空研究所報告 第6巻, 第3号, 1970年9月.
41) 穂坂 衛: "自由形状曲面の理論と設計", 情報処理 Vol. 8, No. 2, pp. 65～72, 1967.
42) 穂坂 衛, 遠藤 誠: "図形の発生, 処理, 記憶", 情報処理 Vol. 6, No. 3, pp. 129～139, 1965.
43) Jayce, J. D. & Cianciolo, M. J., "Reactive displays: Improving man-machine graphical communication", Prroc. FJCC, 1967.
44) Johnson, T. E., "Sketichpad 3-A computer program for drawing in three dimensions", Proc. SJCC, 1963.
45) Johnson, C. I., "Principles of interactive systems", IBM Systems Journal, Vol. 7, No. 304, 1968.
46) Kennedy, J. R., "A system for time-sharing graphic consoles", Proc. FJCC, 1966.
47) Kesselman, M. L., "How do we stand on the big board?", Proc. FJCC, 1967.
48) Kulsrud, H. E., "A general purpose graphic language", Communications of the A. C. M., Vol. 11, Nov. 1968.
49) Lang, C. A. & Gray, J. C., "ASP-A ring implemented associative strvcture package", Communications of the A. C. M., Vol. 11, No. 8, Aug. 1968.
50) Machover, C., "Graphic CRT terminals-characteristics of commercially available equipment", Proc, FJCC, 1967.
51) Mahl, R., "An analytical aproach to computer systems scheduling", Univ. of Utah, Computer Science, UTEC-CSc-70-100, June 1970.
52) 松井 茂: "ディスプレイ装置", 印刷雑誌, 別冊, 1969.
53) Morrison, R. A., "Graphic language translation with a language independent processor", Proc.

FJCC, 1967.
54) Newman, W. M., "A system for interactive graphical programming", Proc. SJCC, 1968.
55) Ninke, W. H., "Graphic 1 : A remote graphical programming", Proc. FJCC, 1965.
56) 大須賀節雄："入出力装置としてのグラフィック・ディスプレイの位置づけ"，エレクトロニクス 1969年11月
57) Poole, H. H., "Fundamentals of display systems", Spartan, 1966.
58) Prince, M. D., "Man-computer graphics for computer-aided design", Proc. of the IEEE, Vol. 54, No. 12, Dec. 1966.
59) Pyle, J. L., "Digital plotting on bussiness", Datamation, July 1967.
60) Ridinger H. J., "The Grafacon man-machine interface", Datamation, July 1967.
61) Roberts, L. G., "Graphical communication and control language", Display Review, 1964.
62) Roberts, L. G., "The Lincoln Wand", Proc. FICC, 1966.
63) Romney, "Computer assisted assembly and rendering of solids", RADC-TR-69-365, Sep. 1969.
64) Ross, D. T., "The AED free storage package", Communication of the A. C. M., Vol. 10, No. 8, Aug. 1967.
65) Ross, D. T., "The AED approach to generalized computeraided design", Proc. A. C. M. National Meeting, 1967.
66) Ross, D. T. & Feldman, C. G., "Verbal and graphical language for the AED system : A progress report", MIT, Project MAC, MAC-TR-4, May 1964.
67) Ross, D. T. & Rodrigues, J. E., Theogetical foundations for the computer-aided design system", Proc. SJCC, 1963.
68) Rovmer, P. D. & Feldman, J. A., "The LEAP language and data structure", Proc. IFIP 1968.
69) Rully, A. D., "A subroutine package for FORTRAN", IBM Systems Journal, Vol. 7, No. 304, 1968.
70) Sackman, H., Erikson, W. J. & Grant, E. E., "Exploratory experimental studies comparing online and offline programming performance", System Development Corp., SP-2687 1966.
71) Sibley, E. H., Taylor, R. W. & Ash, W. L., "The case for a generalized graphic problem solvor", Proc. SJCC, 1970.
72) Siders, R. A. & others, "Computer graphics : A revolution on design", American Management Association, Inc., 1966.
日本語訳 藤沢 忠："コンピュータ・グラフィックス"，産業能率短期大学
73) Sproull,rR. F. & Sutherland, E. I., "A clipping divider", Proc. FJCC, 1968.
74) Stineman, R. W., "Plotting a function of three independent variables", Communications of the A. C. M., Vol. 10, No. 7, July 1967.
75) Sutherland, I. E., "Sketchpad : A man-machine graphical communica ion system", Proc. SJCC, 1963.
76) Sutherland, I. E., "Computer graphics", Datamation, Vol. 12, May 1966.
77) Sutherland, W. R., "The CORAL language and data structure", MIT Lincoln Laboratory TR-405.
78) Symonds, A. J., "Auxiliary-storage associative data structure for PL/1, IBM Systems Journal, Vol. 7, No. 304, 1968.
79) Tally, D., "Automatic plotting in the third generations", Datamation, July 1967.
80) Terlet, R. H., "The CRT display subsystem of the IBM instructional system", Proc. FJCC. 1967.
81) Theis, D. J. & Hobbs, L. C., "Low-cost remote CRT terminals", Datamation, June 1968.
82) Thomas, E. M., "System considerations for graphic data processing", Computers and Automation, Nov. 1967.
83) Thomas, E. M., "GRASP-a graphic service program", Proc. A. C. M. National Meeting, 1967.
84) Van Dam, A. & Evans, D., "A compact data structure for storing, retrieving and manipulating line drawings", Proc. SJCC, 1967.
85) Ward, J. E., "Systems engineering problems in computerdriven CRT displays for man-machine communication", IEEE Trans. Vol. SSC-3, No. 1, June 1967.
86) Warnoch, J., "A hidden line algoriyhm for half-tone picture presentation", TR-4-5, Univ. of Utah, May 1969.
87) Wehrli, R., Smith, M. J. & Smith, E. F., "ARCAID : The ARChitect's computer graphics AID", Univ. of Utah, Computer Science, UTEC-CSc-70-102, June 1970.
88) Wexelblat, R. L., "The MULTILANG on-line programming system", Proc. SJCC, 1967.
89) Zimmerman, L. L., "On-line program debugging : A graphic approach", Computers and Automation, Nov. 1967.

(昭和46年3月31日受付)