44-S 004

図形表示用プログラムの基礎研究

（図形表示プログラムの研究開発）

昭和 45 年 3 月

財団法人　日本情報処理開発センター

この事業は，日本自転車振興会の機械工業資金による「昭和４４年度情報処理に関する調査・研究補助事業」の一部として実施したものであります。

序 に 代 え て

当財団は，情報処理システムにおけるコンピュータ・グラフィックスの応用に関する研究を進めておりますが，この研究報告書はその一環として当財団に設置のグラフィック・ディスプレイとX－Yプロッタを用いて実施した図形表示用プログラムの開発に関する基礎研究の結果と，とくに人間とコンピュータとの情報交換という観点からみたインタラクティブ・グラフィック・システムの開発上の問題点をとりまとめたものであります。

ここに本研究実施にご尽力ならびに御支援を賜わった関係各位に心より感謝の意を表しますとともに，本報告が各方面に利用され，わが国情報処理産業発展の一助として寄与できますよう念願いたす次第であります。

昭和４５年３月

財団法人　日本情報処理開発センター

会長　　難　波　捷　吾

目　　次

1. インタラクティブ・グラフィック・システム

2. インタラクティブ・グラフィックスのソフトウエア

2.1 インタラクティブ・グラフィック・システムの設計基準 …… 3
2.2 インタラクティブ・ジョブ・プロセッシング …… 5
2.3 データ・ストラクチュア …… 18
2.4 ランゲージ・ファシリティ …… 42
2.5 参考文献 …… 53

3. コンピュータ・グラフィクスにおける数学的手法

3.1 PICTURE CALCULUS …… 60
3.2 投象問題 …… 65
3.3 曲線，曲面のFitting と Smoothing …… 74
3.4 隠れ線消去 …… 84
3.5 参考文献 …… 93

4. 三次元図形操作プログラム

4.1 目的 …… 97
4.2 使用機器構成 …… 98
4.3 プログラムの概要 …… 99
4.4 実験例 …… 103
4.5 問題点 …… 104

5. X－Yプロッタの基本パッケージ

5.1 ハードウェアに関して …… 107
5.2 ソフトウェアに関して …… 111
5.3 簡単なアプリケーションの一例 …… 123

あとがき …… 127

1．インタラクティブ　グラフィック　システム

マン・マシン・インタラクティブ（Man Machine Interactive）という言葉が使われだしてすでに久しい。人間と機械が密接に連絡をとりながら共同して仕事をするという意味から，同意語でマシン・エイデド・コグニション（Machine Aided Cognition）という言葉もある。

コンピュータの入力装置，出力装置は従来から人間と機械の情報交換の舞台であった。そしてディジタル情報を扱うコンピュータとアナログ情報を扱うコンピュータとは一般に区別されており，両者の機能を併せ持つものとしてハイブリッド・コンピュータが特殊目的に使われて来た。しかし各種の目的に汎用に使われるものとしては専らディジタル・コンピュータが活躍している。そして従来ディジタル・コンピュータ用の入出力情報は矢張りディジタルが主体であった。勿論アプリケーションによっては，プログラム・コントロールによってグラフィカル（graphical）なアウトプットを出す事は可能であるし，X−Yプロッタでカーブを出力する事も可能であった。

しかし，これらの機能は，専らバッチ処理用の道具として使われ，インタラクティブ・システムとしては一般に不適当である。ある種の処理結果を見て，使用者が更に新しい処理の要求を出す。その結果を見て，更にまた新しい処理要求が生れる……，という様にその場で次々と新しい要求を出し，機械が即時にそれに応ずる，という事になって始めてインタラクティブ・システムと言うことができる。

キーボード・タイプライタを使用した会話型システムというのも代表的なインタラクティブ・システムの1タイプである。しかしこれはグラフィカルな情報を処理するには不適当である。また，たとえそれがディジタル情報であっても，情報量が比較的多い場合は，一字ずつ表示されるという時間的，感覚的制約もあって，出来ることなら，出力情報が一度にパット表示される方が好ましい場合が多い。

CRTディスプレイがディジタル・コンピュータの入出力装置として脚光をあびる様になったのは，以上のべた様な種々の目的を一度に解決する有力な武器と思われたからである。事実インタラクティブ・グラフィックスの活用によって，コンピュータ利用の世界は，大きく新生面を切りひらいた。

航空機あるいは自動車工業におけるCAD（Computer Aided Design），構造解析，グラフィックAPTをはじめ，コマンド・アンド・コントロール・システム，CAI（Computer Assisted Instruction），シミュレーション・システム，技術計算，統計計算のグラフィカル表示，あるいはインフォメーション・リトリーバルのインタラクティブIOとして，さまざまな分野でこのグラフィック・ディスプレイ装置が実用される様になった。

ところが残念なことにこの魅力ある装置もうまく使いこなすには，なみなみならぬソフトウエア・サポートが必要であり，しかも，ハードウェアの機能，アプリケーションの種類によってソフトウエアへの要求も種々様々であり，汎用パッケージ，あるいは汎用言語を作り，個々のユーザの労力を減らすという事もやり難い。

また，ハードウエア，ソフトウエアとも，コスト的にはなかなか高価であり，大衆的に広く利用される状態にはまだ到っていない。

しかし，これらの問題を解決すべく様々な研究，努力が各方面で行われている事は勿論であり，い

づれにせよ今後，グラフィック・システムが，人間と機械のよき仲介者として益々利用範囲をひろげるであろうという事は疑いないところであろう。

2．インタラクティブ　グラフィックス
ソフトウエア

2.1 インタラクティブ・グラフィックス・システムの設計基準

インタラクティブ・グラフィックス・システムを設計するに当って，考慮されなければならない問題点について，整理すると次のようになる。

(1) 装置（特にＣＲＴ）の特質を良く生かすこと

— 人間の直観に直接訴えることのできる，図形を介してマン・マシン間のインタラクションを行うことができるのがインタラクティブ・グラフィックスの本質であるから，その窓となるべきＣＲＴおよびライトペン等の機能は，フルに生かされなければならない。

(2) 使い心地の良さ

— ユーザ・オリエンティッドなハイレベルの言語で，各種アプリケーション・プログラムが容易に作成でき，ジョブの定義，始動等もシステムにガイダンスの機能を持たせる等して，楽に行えるようにすること。

それにディバックの便利さも大きな条件であろう。

また全体を通じて，速やかなレスポンス，信頼性ということも十分考えなければならない。

要するに，ユーザ（アプリケーション・プログラマも含む）がそのシステムをかなり長時間使用してもイライラしないようなシステムにすべきである。

(3) 汎用性

— この汎用性というのは非常に難しい問題ではあるがこれを実現するアプローチがどうであれ，個々のアプリケーションに依存せず，どのような仕事にも充分に効果のあるサポートができ，必要に応じて任意の巾広い機能が利用できるようにしてやることは重要なことである。

(4) 拡張の容易さ

— グラフィック・システムはアプリケーション分野の広範さに応じて，そのサポート・システムも多種多様な機能が求められる。それらはとても一時に全ての要求を満足するようなフル・セットを提供しがたい規模であり，サブ・セットから作りあげて順次機能を追加して行くのが妥当な開発へのアプローチであろう。

この時，新たに開発されたサブシステムを従来のシステムに追加していく拡張の容易さは十分考えられなければならないことである。

(5) システムのシンプルさ

— 必要以上に複雑なシステムは，スループットを低下させ，またシステムのメンテナンス，バージョン・アップを困難にする。どのようなシステムにも共通して言えることだが特にグラフィック・システムではややもすると，データ・ストラクチュア，ジョブ・コントロールなど複雑に過ぎるものを選択しがちになりそうであるが，なるべくシンプルにまとめる必要がある。

(6) 使用効率の向上

— ディスプレイ装置はまだ非常に高価であるし，人間が考える時間が多い利用方法が殆んどである。

従って，マルチ・ジョブ・オペレーションを可能にする等してシステムの使用効率を高める必要がある。

これらの条件は非常に概念的であり，具体性を欠いているが，この条件を十分に意識しつつ，インタラクティブ・グラフィックス・システムを設計する際に，そのファンクションを次の3つに分けて各々について次節から言及したい。

1. インタラクティブ・ジョブ・プロセッシング
2. データ・ストラクチュア
3. ランゲージ・ファシリティ

2.2 インタラクティブ・ジョブ・プロセシング

2.2.1 インタラクティブ・グラフィック・システム

Computer Graphicsというのは，第一章でも述べたように，

「線，曲線，点等のグラフィック・シンボルを使って，コンピュータと話をする人間の概念」

「人間と機械の会話におけるもう一つの自由」

等と呼ばれている。

このようなコンピュータ・グラフィックスを用いた，実用的なアプリケーションを考えると，次の二つの要素が要求される。

(1) 大容量のオンライン・データ・ベースに対する高度な処理機能

(2) 人間と機械の間の多数の情報交換を行う

アプリケーションの種類によって，データ・ベースの大きさは，千差万別であろうが，一般的には，かなりの容量のデータ・ベースが存在すると考えられる。

高速な情報の検索，処理はグラフィックの1つの大きな要素となる。

コンピュータ・グラフィックスにおけるマン・マシン・インタラクションは，従来のコンピュータのプログラム・コントロールの概念を大きく変えた。

いまや，プログラムの流れを，コントロールするのはコンピュータではなく，人間だと言ってもいい。

まず，インタラクティブという言葉を少し考えてみよう。

これは「相互作用を持つ」という意味であるから，システムの立場からみれば，

「ある入力と，それに対するシステムの応答」[*1] として定義される。従って，あるシステムに対してある作用（action）を入力し，その反応（reaction）が応答として返ってくるような形で進められるような仕事をinteractive modeのjobと呼ぶ。

具体的な例として，今ライトペンを用いてCRT画面に一本の伸縮する（rubber－band）線分を描く動作を考えてみると，これは次の5つの手順からなっている。

(1) ペン・トラッキングのためのボタンを押す。

(2) 線の始点までペンを追いかける。

(3) 始点を固定するボタンを押す。

(4) 終点までペンを追跡する。

(5) 終点を固定するボタンを押し，終了。

この手順を，3個の状態を持つ有限オートマトンとして考えれば，図1の如く示される。

この，人間と機械のやりとりについて，ロッキード・ジョージア社のM. D. Prince氏は，次

のように述べている。

「人間とコンピュータとの統合された働き（Man－Computer Synergism）は，人間とコンピュータが自由に情報の交換を行うことによってはじめて可能になる。すなわち，人間とコンピュータとの間の障害が取りはずされ，"リアル・タイムでの会話"が行なわれるようになることである。この人間とコンピュータとの会話は，コンピュータのスピード，プログラミング技法および周辺入出力機器の発達などで近年可能になってきている。"会話"はタイプされたメッセージ，数字あるいはCRTにディスプレイされたシンボルなどの型で行なわれたり，人間およびコンピュータの双方が理解できる他の適当な型で行なわれたりしている。

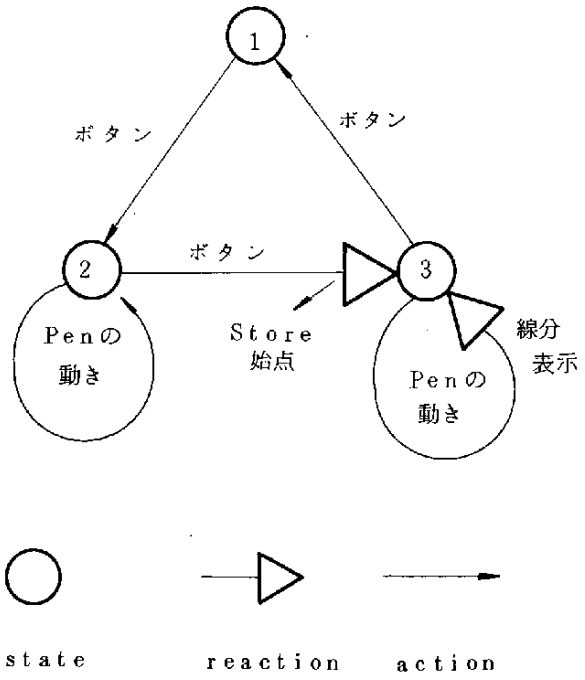


図 1

また，グラフィック・ディスプレイを用いた会話型システムに対する要求事項としてIBMのC. I. Johnson は，より詳しく次の11項目を挙げている。*2

1. システムとのやりとりは単純であること —
   コンピュータ・プログラミングの知識のないアプリケーション・ユーザに対しても十分単純でeasy－to－use なコマンド・ランゲージを備えること。
2. コマンド・ランゲージは「寛大」であること。
   コマンドにエラーが起った時は端末ユーザが再入力することを許すこと。
3. エラー時はその診断ルーチンを働かせるのみで，そのジョブをアボートしたりしないこと。システムで同時にランしている他のユーザに影響を与えたりしないこと。
4. 端末からプログラム・エラーを診断する機能を有すること —
   アプリケーション・プログラマは，エラーをオン・ラインで修正しソース・プログラムを変更し，再びコンパイルし，再実行ができるようになっていること。
5. コマンド・ランゲージは，ソース・プログラム・ファイルや，プログラム・ライブラリ，データ・

*1. A System for interactive graphical programming W.M. Newman '68 SJCC

*2. IBM Systems Journal Principles of interactive systems, C. I. Johnson

ファイルを創成し維持することのできる基礎的な機能を提供すること。

6. アプリケーション・プログラマがディスプレイ・コンソールを用いて，ソース・ファイルから，オブジェクト・コードにアセンブルしたり，コンパイルすることができ，かつ出来あがったオブジェクト・ファイルのいくつかをリンクして1つのプログラムとし，それを実行できること。

7. プログラマあるいはアプリケーション・ユーザのファイルは他人から保護してもらうことができるが，それと同時にデータ・ベースあるいは共通ファイルとして他のユーザに対しても"shared"として区別して用いることができること。

8. バック・アップとか再起の機能（特にファイルに対して）は非常に重要である。

9. アプリケーション・プログラマあるいはアプリケーション・ユーザがディスプレイ・コンソールから新しいコマンドを定義し，システムを拡張することができること。

10. ディスプレイ・オリエンティッド・システムへの特別の要求は，ユーザ・コマンドのヒストリーの印刷とコンピュータの動作の結果を"hard－copy"として，とることができること。

11. 会話型のシステムは，ユーザがデータを定義したり，プログラムをcallしたりすることをユーザの要求に応じて許してやらなければならないので，主記憶のダイナミックなアロケーションの機能を備えていなければならない。この機能の使用法は簡単であり，アプリケーション・ユーザにとっては"透明"（みえない）でなければならない。少なくともリソースのすべてが欠乏するまでは，……

プログラムは，システムに呼び込まれたときにダイナミックにリンクされなければならない。

これらの要求は，普通会話型のジョブで用いられているテレタイプとかタイプライタ端末でも大体同じ様なことが言えるわけだが，ディスプレイ・コンソールは結果の表示が瞬間的であり，ユーザはよりインタラクティブな効果を期待する傾向があるので，これらの要求も一そう強張される。

2.2.2 インタラクティブ・グラフィック・コンソール

コンピュータ・グラフィックスにおいて，インタラクティブな機能は必要欠くべからざるものであることは前節で明らかになった。ここでは，グラフィック・ディスプレイをユーザとコンピュータの間のはし渡しであるコンソールとしてとらえた場合に，どのような機能が必要になるかを考える。（以下，IBMのE．M．Thomasによる「System Consideration for Graphic Data Processing」より引用）

グラフィック・コンソールには，ディスプレイの作動やユーザプログラム，制御プログラムによるユーザのインタラクションを司さどるために色々の操作が要求される。

これらは，インタラクティブな機能，レスポンス，エラー時の回復，それに信頼性である。

(1) インタラクティブな機能

グラフィック・コンソールのインタラクションに望まれる制御プログラムの機能は，大別すると，次の3つが考えられる。

(イ) ジョブの始動

(ロ) 非同期処理

(ハ) リソース割当

グラフィックな処理における，ユーザからシステムへのアクセスはグラフィック・コンソールを介して行なうべきであり，従来のバッチ・ジョブにおけるものとは違っていなければならない。すなわち，グラフィック・コンソールは，プロブレム・プログラムのコントロールとインタフェイス（入出力デバイスとして）の両方に用いられるべきである。

(イ) ジョブの始動に関して

○ ジョブの始動はごく自然に，会話型で行なわれるべきである。

○ コンソール・ユーザは，ジョブ制御文の詳細等は知っている必要がなく，自分のプログラムやデータに関する名前あるいは識別情報を入力するのみでよい。

○ あらかじめ定義されているジョブ制御文のいろいろな機能が，ある問題の実行中，必要なら何時でも使用できること。

○ ジョブが始動されたなら，すぐに実行スケジューリングに乗ること。ユーザはグラフィック・コンソールの前で自分の仕事が始まるのを待っているのだから。

(ロ) 非同期処理に関して

プロブレム・プログラムが始動すると，グラフィック・コンソールはユーザとプログラム間の通信に使用される。

前述したように，プログラムの実行中，ある種のシステム・レベルの機能がユーザから望まれることがある。このことを非同期の機能と呼ぶ。というのは，ユーザはこれらを自分のプロブレムの処理中に，あらかじめ定義されていない任意の時点でcallするからである。

従ってプロブレム・プログラムのレベルから制御プログラムのレベルに戻るような機能が必要になる。このために，言わば簡単なディスク・カルキュレータや中規模の設計問題にでも，ユーザからのシステム・コールを許すためにsave，restoreの機能がなければならない。

又，システムはすでに存在しているデータ・セットに対し修正を行ったり，新しいデータ・セットを創成したり，その創成を途中で停止したりできるべきである。

又，何か仕事をやっていてうまく行かなくなった時にコントロールを取り戻す為の，プログラム・ディバッグの機能が必要である。

例えば，ユーザは以前うまく行った，あらかじめ知っている状態にプログラムを戻したい時があるから。

（この非同期処理に関しては後のランゲージの項でもふれている）

(ハ) リソースの割当について

大型計算機が複数個のコンソールで共同利用されている時，コントロール・プログラムは，CPU時間，記憶場所，ファイルのエリア等のシステム・リソースの割当を管理する必要がある。

CPU時間の割当はあるコンソールが長時間の計算を必要とする時，他のコンソールに対してロックをかけたりしないようにする為に必要である。

記憶場所の割当は，ジョブの実行に必要なエリアを，ジョブの始動時に確実に確保できるようにする為に必要である。

ファイル・スペースの割当はユーザ・プログラムが実行中に創り出すデータの為に必要になる。また同一データを複数のユーザで共同に利用できる機能も必要である。

(2) レスポンスについて

(イ) レスポンスの種類について

ユーザがコンソールで何か操作を行った時には，いつもある種の返答があるのを期待している。このユーザの動作とコンピュータからの応答の間の時間がレスポンス・タイム（応答時間）と呼ばれるものである。

グラフィカルな操作に関するレスポンスは次の3つに大別することができる。

i） 即時の応答

（例えば，メニューの選択とか，要素の選択，データの登録などである）

ii） 数秒～10 secの応答

（新しいプログラムを探しているとか，図面の描き直しとか）

iii） もっと長い時間が許される場合

（新しいデータあるいは修正したデータによるアプリケーション・プログラムの処理時間でもちろんこれは機械の処理時間とアプリケーションの種類に依存する）

これら3つのタイプの速応答は，ごく平凡な計算処理を要求するだけである。しかし他のタイプのものはより多くの計算能力を必要とする。

グラフィック・アプリケーションの多くは大容量のデータ・ベース，大きいプログラム，一つのレスポンス対する長い計算，あるいはこれらの組合せを扱うのに大型のコンピュータを必要とする。この大型コンピュータはコンソール・ユーザが何か操作を行った時のみ使われるだけである。

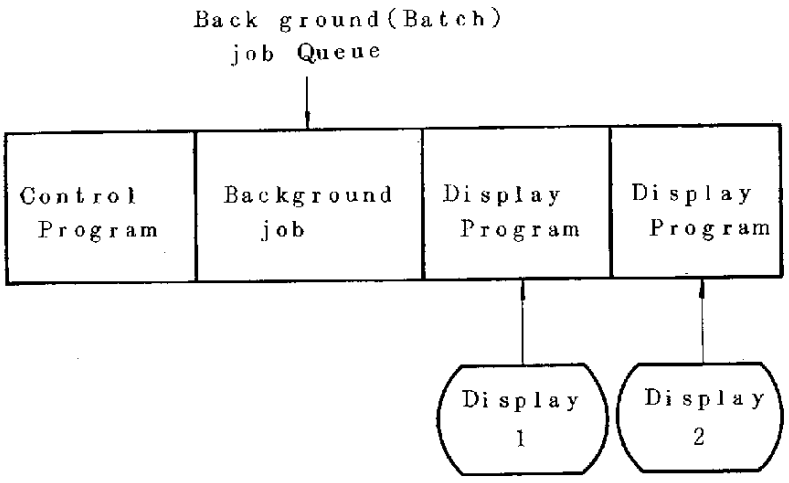


図2 メイン・コンピュータの共同利用

このように大型コンピュータが占有されたとすると，あるアクションの間（ユーザが考えている時間等）の時間は無駄になり，非常にコスト高になる。

このようなことから，複数のコンソールで，あるいはコンソールとバック・グランドのバッチ・ジョブで分割して利用するシステムの型が望まれた。（図2）

(ロ) 制御プログラムの機能

ローカルに接続されたグラフィック・コンソールでは前述した3種類のレスポンスのすべてをメイン・コンピュータで扱うことが必要である。

このレスポンスのために，制御プログラムでは次のような機能が必要となる。

ⅰ） ユーザからの信号を処理するためのスケジューリング・ルーチン

ⅱ） プログラムのどの部分でも効率よく呼びだすことができる。

というのは会話型のプログラムは大抵，使用可能なエリアより多くのコア・スペースを必要とするから。

ⅲ） 大きいデータ・ベースの必要な部分を必要な時間内にアクセスすることができること。

しかし，ローカルに接続されている場合は，大抵この時間は小さく，さほど重大に考える必要はない。

グラフィック・コンソールがメイン・コンピュータから離れている場合には，メイン・コンピュータとリモート・ロケーションとの間で，適当な仕事の分担が必要になる。

即時のレスポンスを行うためには，どの機能をリモート側で処理するかを，夫々の目的に応じて選ばねばならない。

(3) エラー回復と信頼性について

インタラクティブなコンソールを使うことによって，ユーザはシステムの信頼性に，より気を使うようになる。

バッチで処理する場合は，距離的にシステムの近くにいるという事もあり，システムの異常が直接わかるので，あまり不安を感じないで仕事ができる。しかし，リモート・ターミナルからの仕事は一般にメイン・コンピュータの状態を密接にとらえにくいので，エラー・オペレーションに対する反応や，システムの信頼性に対してより一層の関心を払う様になる。

ユーザの信頼を得ることは，有用なシステムをつくる上で最も重要なことの一つである。しかし不幸にしてエラーを起してしまった場合は，なるべく早く回復することが必要である。

エラー回復は，プログラムとデータの両方に関係してくる。

一人のユーザの仕事は他のユーザの仕事に悪影響を及ぼしてはならない。

もしあるプログラムが異常終了したとしても，他のユーザのジョブを停止させたりせず，他のジョブの再スケジューリングを行うのみでなければならない。

このことは異常終了がプログラムによるものであっても，ハードのエラーによるものであってもどちらでも考えられなければならないことである。普通，グラフィック・コンソールでの仕事はかなり長時間を要する。その仕事が完全に終了するまえに，何かが起った場合，今迄やった仕事を全

部やり直すようなことを未然に防ぐような方法が必要である。

このため，プログラムやデータのセーブ（save），リストア（restore）の機能がバック・アップとして用いられる。

2.2.3 システム構成上の問題点

(1) 小型システムと大型システムについて

グラフィック・コンソールに小型のコンピュータを用いて設計問題の解決に成功している例も数多くある。アプリケーションが大きいデータ・ベースとか，大きな計算を必要としないならば，小型コンピュータで充分である。

小型システムはコストも安価であり，それ専用にあつらえた状態で使用されるので効率もよい。

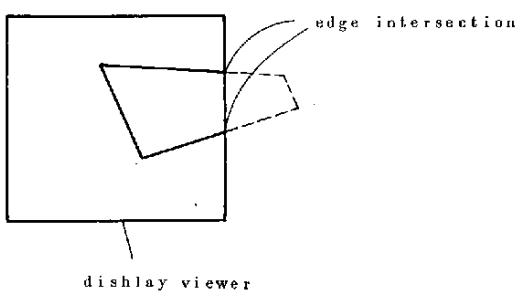
この小型システムを，大容量データ・ベースへのアクセス，またはより高度の計算，あるいはその両方の目的で，大型コンピュータに接続して利用すると，より有用なシステムになってくる。

この場合，小型システムのコンソールは中央の大型コンピュータからリモートに位置し，画面の再生，図形的な修正，小量の計算等は小型システム側で分担する。

プログラム制御の能力を持っているターミナルを利用することにより，より興味あるシステム体系を生みだす。

(2) システム・リソースの配置

図形処理プログラムの多くは，大型計算機の高速の処理能力，大量の記憶装置スペースを要求する。グラフィック・CRT装置への図形表示は，特殊なCRT装置（例えばストレッジ・チューブ方式のもの）を除いては，普通の入出力装置と異って，表示される図形データは，ディスプレイ・コマンドの列としてコア（又はバッファ）にあり，制御装置がこのデータの連なりを巡環しつつ，トレースしてCRT・ビームの駆動を制御しているため，一度に表示すべき図形データは，そのCRT表示の間を通じて記憶装置を専有することになる。これのみをとってみても，かなりのリソース要求になるが，なかんずく，グラフィック・CRT入出力においては，多種にわたる外部割り込み制御が必要であり，この処理に要するコア・スペース，各種機器装置の増加は相当なものとなる。すなわち，ライトペン，ファンクション・キー，ダイアル，キャラクタ・キーボード等の入力手段や，表示データのエッジ・インターセクションは，ハード的な外部信号割り込みと対応し，その処理は，プログラムにゆだねられるのが一般的な方式である。



さらに，複雑な図形処理のプログラムでは，データ・ストラクチュアの蓄積と操作が要求されるが，この場合には，２次記憶上の大量の記憶スペースと，データの主記憶 ↔ ２次記憶のひんぱんなトラフィック・コントロールが要求されるであろう。これら大量の，ストレッジ・リソース要求に加えて，グラフィック・ＣＲＴを媒介として，マン・マシン・インタラクションを含むプログラムは，ＣＲＴオペレータの要求に対する，素速いレスポンスを実現するために，高速のＣＰＵリソースがまた要求される。

インタラクティブ・グラフィック・システムにおける，リソース要求の特徴を列挙すると，次のようになろう。

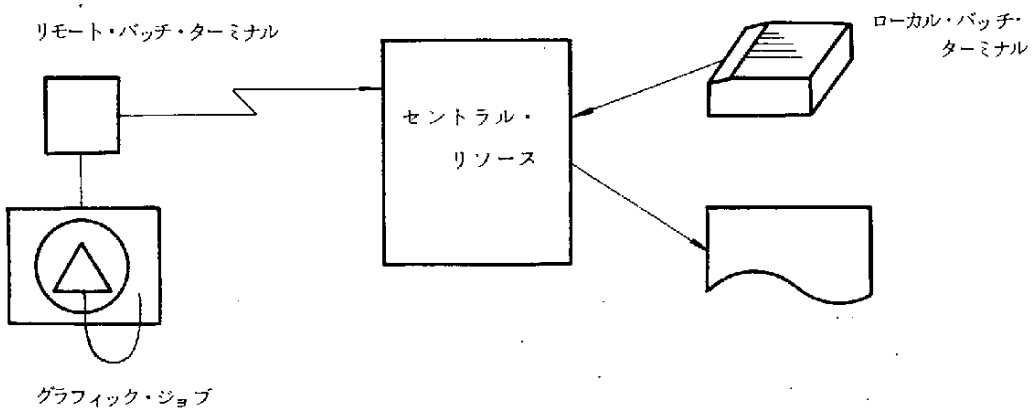
① 大量の主記憶，及び２次記憶スペース

② 高速のＣＰＵ，及びデータ・トラフィック・メディア

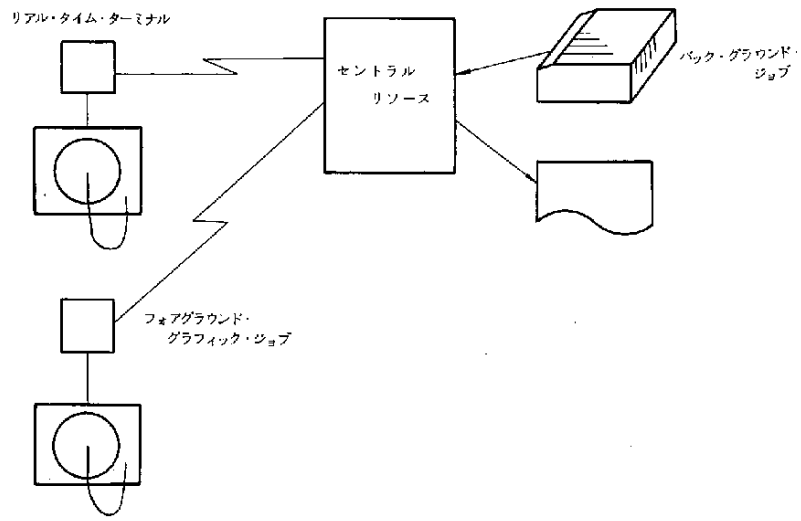
③ ＣＲＴ・オペレータの応答を待つ間は，表示中のディスプレイ・コマンド列用のバッファを除いた，大部分のリソースが不用

これを要約すると，「大型計算システムが必要だが，単一のグラフィック・プログラムにこれを占有させるのは，無駄が多い。」と云うことになる。グラフィック・システムを経済的に妥当なものとする条件の一つは，適宜なリソース・シェアリングを伴なった，マルチ・プログラミングの実現である。

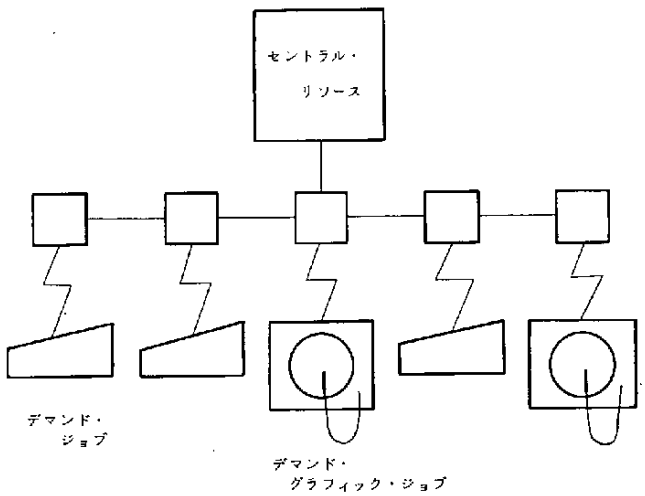
実際に，多くの既存グラフィック・システムでは，これをマルチ・プログラミングの内に組み込んでいる。その一つの例は，２バッチ・システムである。２バッチ・システム下では，グラフィック・プログラムを，優先度を高くしたリモート・バッチ・ジョブとして，また，非グラフィカルなプログラムは，通常のローカル・バッチ・ジョブとして投入される。この他に，グラフィカル・プログラムを，リアルタイム・ジョブとして組み込むシステムや，ＴＳＳエンバイロンメント下で，デマンド・ジョブの一つとして組み込むシステム等がある。



２バッチ・システム



リアル・タイム・システム



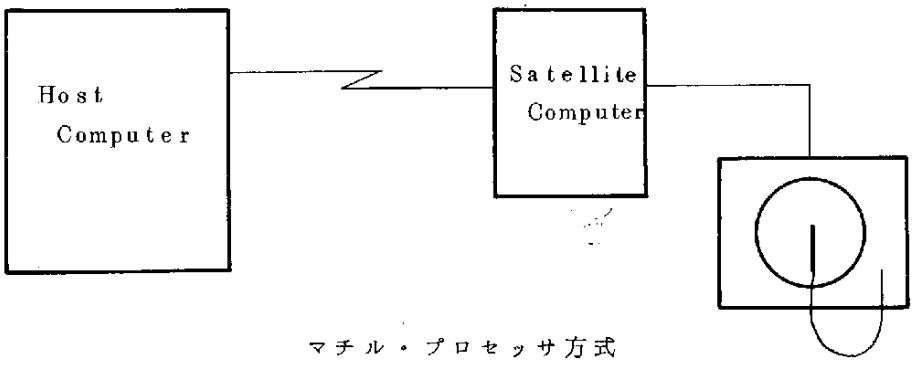
タイム・シェアリング・システム

マルチ・プログラミングは，グラフィック・システムのコスト・パフォーマンスを上げる一つの手段ではあるが，必ずしも良いことずくめではない。難点の一つは，マルチ・プログラミングの下では，オーバ・ヘッドのある程度の増加がどうしてもさけ難いことである。マルチ・プログラミングも，グラフィック・ジョブが一時には一つきりで，他は非グラフィカルなジョブのみであれば，ジョブ間に優先度の差を設け，非グラフィックなジョブに犠牲を強いることで，問題はある程度解決する。通常のバッチ・ジョブであれば，オーバ・ヘッドの多少の増加は，与える影響が少ないであろう。

複数のグラフィック・ジョブが，複数のグラフィック・CRTターミナルから呼び出され，マル

チ・プログラミングの下で同時に競合し得るようなシステムでは，問題がやっかいになる。大規模なリソースを要求するジョブと，リアル・タイム・レスポンスを要求するグラフィック・ジョブがかち合った時，システムは最良の場合でも，各々にその半分の能力しか提供できない。悪くすると，システムは両者間のトラフィック・コントロールに忙殺されて，本来の処理 ― グラフィック・ジョブの処理も ― は少しも進行しないといった，いわば，"ともぐい"の状況をかもし出す恐れすらある。

この恐慌をさけるための手段として，グラフィック・プログラムに発生する，多くのインタラクション処理を，なるべく各々のＣＲＴ端末に分散させ，センタ・リソースへの負荷を最少にくい止める方法がある。このようなシステムは，センタ・リソースをhost－computer（またはmain system），ＣＲＴ端末をsatellite－computer（またはintelligence－terminal Subsystem）と呼ぶ，マルチ・プロセッサ方式として実現される。



マチル・プロセッサ方式

しかし一方，マルチ・プロセッサ方式は，システムの肥大化と，プロセッサ間のデータ・トラフィックの必要から，ますますグラフィック・システムのコスト増をもたらす恐れがある。また，マルチ・プロセッサ方式では，両プロセッサに処理をどの程度の割り合いで分担させるか ― つまり，ロード・シェアリングが問題となる。端末側に多くの処理を望めば，satellite－computerもかなりの機能を持ったリソースが必要になり，コスト増加をもたらす。端末側の処理を，ごく単純なものにだけ留めれば，端末で発生した割り込みのほとんどを，host－computer側に伝え，そちらで応答作成の処理をして再び返送を行なう必要から，システム全体のパフォーマンスは低下する一方，両プロセッサ間のデータ・トランスミッション・レイトが上がる。これは，もしＣＲＴ端末が通信回線等を経由して，かなりの遠隔地に置かれるような場合には，軽視できない問題である。

グラフィック・システムにおける，リソース配置の選択は，いずれにせよ一概にその優劣を決し難く，一般的なbestway はないとみなければなるまい。個々のシステムに課せられた諸条件が，ケース・バイ・ケースで考慮され，その中で最優先の条件を良く満足するように努めるしかなさそうである。

(3) ロード・シェアリング

インタラクティブ・グラフィックス・システムを，メインシステムとサブシステムとしての構成で考えた場合，ジョブの分割には以下のような数多くの問題を考慮せねばならない。

イ．サブシステムの大きさ

ロ．メインシステムとリモート・サブシステム間での有効データの割合

ハ．データ・ベースの大きさ

ニ．多種多様の応答に必要な計算処理能力

ホ．ユーザのアクティビティ（活動範囲）

制御プログラムが多種のレベルのレスポンスを処理したり，プログラムをコールしたり，データをアクセスする際の能力は，ジョブの分割をどのようにするかを決定する時に，非常に重要な役割を演ずる。

過去のシステムでは，すべてのオペレーションがメインシステムによって制御されており，データはグラフィック装置とコンピュータとの間を通り過ぎる「もの」にすぎなかったが，サテライト・コンピュータを備えたシステムの場合，このサブシステムは，単なるグラフィック装置と違い，より大きな能力を有するので，何かある種の仕事をやらせることが望まれる。従ってこの場合は単なるデータのみならず命令も転送されなければならない。これらの転送は一般に非同期に起るので，制御プログラムは，複数個のプログラム間の非同期通信を許すような機構を備えている必要がある。

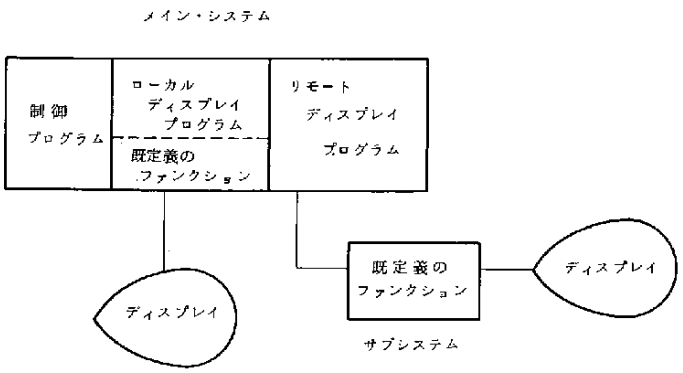
(4) サブシステムのプログラミング

小型のグラフィック・コンピュータあるいはサブシステムは，次のような三種類の方法でプログラミングすることができる。

1. 主制御装置でローカル制御装置をシミュレートするような制限された方法

この方法は，コンパティビリティの利点はあるが，サブシステムとしての力を発揮させる利点はない。

サブシステムがメインシステムからリモートに設置されている場合は，応答特性の面であまり望ましくない。



（dosed サブシステム）

2. "クローズド・サブシステム"としてプログラムする方法

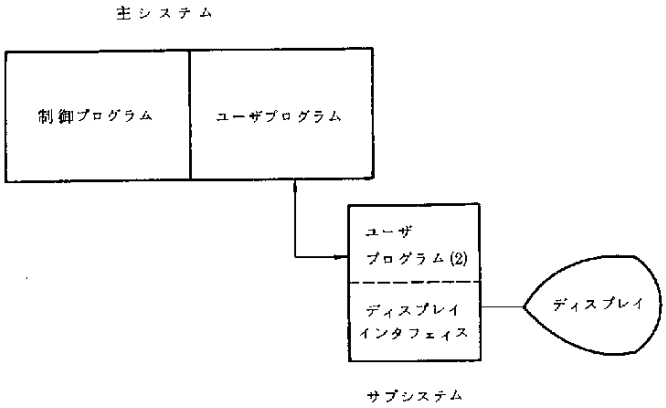
この場合は，前もって定義された機能の集合としてサブシステムがつくられる。

メインシステムは，サブシステムに一つあるいは一連の既定義のファンクションを実行させるよう命令する。

3. オープン・サブシステムとしてあつかう方法

ユーザは，プログラムを2つの部分に分けてつくり，その1つはメインシステムで実行され，1つはサブシステム側で実行される。このことは，ユーザが2つのプログラム間のコントロールと，コミュニケーションが遂行できるようにユーザの位置をあげているわけである。

この方法は，ユーザ・プログラムがかなり複雑になるけれども，それだけにサブシステム側で遂行されるファンクションにより多くのフレキシビリティを許すことになる。



（オープン・サブシステム）

(5) ジョブ・コントロール

サブシステムが使われている時は，ジョブの制御はより多くの事柄を考えなければならない。

サブシステムを，ディスプレイ制御装置同様の使い方，あるいはクローズド・サブシステムとして（前節(1)，(2)）の使い方をしている場合のサブシステムはメインシステムに全く支配され，この場合サブシステムのプログラムはサブシステム側で独自にロードされるか，メイン・コンピュータ側から送られてくる。

最も複雑なジョブの制御というのは，オープン・サブシステム（前節(3)）— この場合は，ユーザはジョブの1部をメインシステム側に，残りの部分をサブシステム側に持つことになる — の時におこってくる。

例えば，リモート・コンソールに対するジョブの始動は，できる限りローカル・ユーザに対するジョブの始動と同じ様でなければならない。

オープン・サブシステムというのは，それ自身で多くの操作を行えるし，又，助けが必要な時のみメインシステムをcallするようになっているので，より自立的である。

サブシステム・エンバィロンメントにおけるデバイスのアロケーションの方法はいくつかある。

その第1は，最も単純な扱いで，サブシステムをある一つのジョブに占有される一つのデバイスとして扱う方法である。

この場合，サブシステム側にそれ自体の1種の制御プログラムを持ち，そのサブシステムに接続されているすべてのデバイスをコントロールする。このモードでは，制御プログラムはサブシステムを単なる一つのデバイスとして扱っている。

第2の方法 — より複雑な — は，サブシステム内のデバイスを別々に扱うもので，それらのデバイスは直接メインから各々独立にアロケートされる。この方法を実現するための主制御プログラムは，より高度のレベルのアロケーション・メカニズム（サブシステムを経由してデバイスの接続を行う）が必要になってくる。

いづれにせよ，一般にメインシステムとサブシステムとは異った命令体系を持ち，必然的にデータ表現も異ることが考えられるので，データ変換を含めた両者の間の基礎的なコミュニケーション・ルーチンのセットが必要となる。

2.3 データ・ストラクチュア

2.3.1 インタラクティブ・グラフィック・システムとデータ・ストラクチュア

グラフィック・アプリケーション・プログラミングでは一般に，大量のデータの，その内容間の関連に基礎を置いた，検索，処理が必要とされる。もちろん，これも場合によりけりで，たとえば，単純な図形出力のみを目的としたアプリケーションでは，そうした必要性はまれかもしれない。そのようなアプリケーションでは，図形データの"図形的性質"に着目，評価するのは，人間の側だけであって，プログラムの側は，その内容には関知せず，個々バラバラなデータの集まりとしての処理を行うのみで，その出力がたまたま人間にとって意味のある図形であると云うに過ぎない。

そうしたアプリケーションはまた，マン・マシン間のインタラクションに乏しいシステムとみることができる。そこでは恐らく，オンラインの情報交換は，マシン → マンの一方向のみで，かつ一過性のものであろうから。

われわれが論議の対象として想定しているのは，そのようなシステムではない。ライトペン，ファンクション・キー，その他の新らしい装置をそなえた，グラフィック・CRTを用いる"インタラクティブ・グラフィック・システム"である。"インタラクティブ"の定義については，2.2節で述べた通りである。

よりインタラクティブなグラフィック・システムにおいては，CRTコンソールを前にしたオペレータ（それは同時にユーザでもある）は，CRT画面上の出力情報（それは，ある場合には，自分の指示に対する応答でもある）を評価しながら，より好ましい状態，あるいは，異った処理へと，プログラムを導くべく指示を与える。そこではたとえば，一度表示した図形が変更を受け，あるいは，その部分が削除，追加を受けるかもしれない。プログラムは，ユーザの指令によって，あるいは，プログラム自身の処理過程に従って，動的に変化をとげる図形を扱えねばならない。動的変化をとげる図形の処理にはそれなりの情報蓄積が要求される。たとえば，動的に変化しうる一つの3角形に関する記憶情報は，3頂点の座標データ以上のものを含むことになろう。3つの直線分がどのように結びついているか，そして，他の図形との関係がどうであるか，…… というような情報を。

また，図形に名前をつけ，図形データをシンボリックに参照できるようにするには，それなりの付加情報が必要である。こうした情報群は，なんらかのストラクチュアを構成する。このストラクチュアは，構成要素の内容間の関連に基礎を置くであろう。構成要素間の内容の関連に基礎を置いたデータ・ストラクチュア — Associative Data Structure — のグラフィック・システムにしめる比重は，非常に大きなもので，これは，IRシステム一般と，共通の課題でもある。

これまでに，多種多様な目的，手段をもって各種のデータ・ストラクチュアが考案されてきた。それらの内，あるものは特定のアプリケーションに深く依存しており，汎用性はない代り，ある限られた目的を実現するには便利で効率が良い。一方，それとは逆の例もある。汎用性の故に小回りがきか

ず，効率が犠牲になっている場合もある。

各種のシステム例を概観してみると，たとえどのような目的，手段を基としても，アソシエーティブ・データ・ストラクチュアをこしらえるのが，どれ程やっかいな仕事になるかゞわかる。…… それは，計算機の記憶装置に，多次元空間の情報を表現することの困難さに他ならない。あるアプリケーションにとって，その図形情報をストラクチュアに編成することは，しばしば，アプリケーションのかゝえている"問題"の解決自体よりも，はるかに面倒な仕事となっている。図形処理を志ざすアプリケーション・プログラムの全てが，このやっかいな作業を，個々のプログラミングの内に課せられるとすれば，図形処理プログラムの作製効率は，きわめて非観的なものとなろう。アプリケーション・プログラマにとっては，その本来の目的である図形データの操作や入出力ができるだけ容易に行えることが望ましく，図形データの内部的な記憶形式，構造がどうなっているか，その操作にどれ程複雑な処理過程が必要とされるか，といった問題にわずらわされずに，従来の非グラフィカルなアプリケーションに要したときして変らないプログラミングの労力で，図形データ処理が行えるのでなければ，コンピュータ・グラフィックスの将来は，ＣＲＴ画面程明るくも，可視的でもない。充分に強力なデータ・ストラクチュア・ファシリティ ― 内容間に関連のある，任意の形式，内容，量を持ったデータをストラクチュアに編成し，その動的操作を可能にする強力なファシリティ ― の出現が期待される所以である。

2.3.2　データ・ストラクチュアとプロブレム・モデリング

汎用性をめざした，各種データ・ストラクチュア・ファシリティの基本的なパターンについて整理しながら，問題点を洗い出してみよう。

グラフィック・システムで用いられているデータ・ストラクチュア中で，最もプリミティブなものは，単純な一次元配列，あるいは二次元配列に代表される。これはたとえば，ＣＲＴビームの位置を，Ｘ，Ｙ座標で表現する，ディスプレイ・コマンド列などの場合に用いられる。

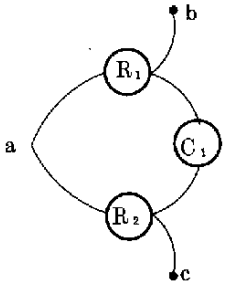
Control	X	Y
C_1	X_1	Y_1
C_2	X_2	Y_2
C_3	X_3	Y_3
⋮	X_4	Y_4
⋮	⋮	⋮

このようなデータは単純なだけに，ＣＲＴビームに対する制御情報としては，最適であろう。もしハードウエア的に可能であれば，ドットの座標のみでなく，直線，円，文字等を指示する情報を加えても良い。これらの指示がＣＲＴ表示に際して有効か否かは，ＣＲＴ制御装置に，ライン・ジェネレータ，カーブ・ジェネレータ，文字ジェネレータ等の機能がそなわっているか否かに依存して

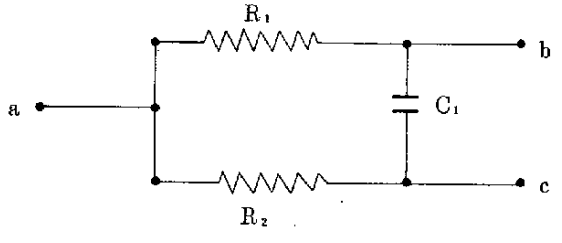
いる。

しかし，こうしたデータ群は，前述したような一過性の（いゝかえればスタティックな）図形表示には充分であるが，それ以上の処理を裏づける情報はなにも持っていない。しかも，このデータ形式は，個々のグラフィック・ＣＲＴ，及び制御装置のハードウエアと，深く密着している。

なんらかの問題をストラクチュアにモデリングする場合 — そのストラクチュアは，マシン・リソースからは独立であることが望ましい。さらに，あるアプリケーションにとって，問題をストラクチュアにモデリングするということと，その問題の図形的外観 — ＣＲＴ画面上での — を与えるために，データ・ストラクチュアを編成することとは，必ずしも同義ではない。たとえば，回路解析のアプリケーションについて考えてみよう。回路のＣＲＴ画面表示は，普通，抵抗器，コンデンサー，リレー，その他の部品の記号を結線した回路図として行なわれるであろう。このような，回路図表示のためのデータ群は，各部品記号の幾何学的形象，結線の位置，角度，その他諸々の図面的配置情報から構成される。しかし，一方，この回路の特性を解析するプログラムにとっては，回路部品の図面的配置，形象などの情報は，ほとんど意味がない。各部品の種類，特性，及び部品間の結線 — つまり，トポロジカルな関係 — についての情報さえ得られゝば良い。



回路解析に必要な回路情報



図面的な回路情報

上記２目的に，各々最もオプティマイズされたデータ・ストラクチュアが，同じ編成となることはないであろう。両方の目的による必要性を共に満たすデータ・ストラクチュアは，むしろ各々の効率を落としめる危険がある。

こうした事情から，グラフィック・アプリケーションのデータ・ストラクチュアを，ＣＲＴ画面上に外部表示するためのデータ・ストラクチュア — ディスプレイ・データ・ストラクチュア — と，問題を内部的にモデリングするためのデータ・ストラクチュア — プロブレム・データ・ストラクチュア — とに分割するアプローチが生まれた。これは，"ダブル・データ・ストラクチュア"とか，"マルチ・レベル・データ・ストラクチュア"と呼ばれている。このアプローチにおいては，ディスプレイ・データ・ストラクチュアは，ＣＲＴ，制御装置，及びオペレーティング・システムなどの，System environment に依存しているが，モデリングされた問題からは独立であることができる。一方，プロブレム・データ・ストラクチュアは，その構成内容が，特定のアプリケーション分

野に依存する代りに，System environment からは独立であることができる。

マルチ・レベル・データ・ストラクチュアの出現はまた，それらを操作するプログラムを，機能別，レベル別に分割せしめ，プログラムのモジュール化促進に一役買った。これらはあいまって，CRTを介した入出力の複雑なステップを代行する，汎用のグラフィック・サブルーチン・パッケージ出現への道を招き，図形処理のアプリケーション・プログラムを容易にせしめた。そのようなサブルーチン・パッケージは随所で開発され，実用に供されている。

しかし，こうした，グラフィック・サブルーチン・パッケージの出現で，データ・ストラクチュアの問題が全て片づいたわけではない。それは単に，CRT経由の図形情報入出力に際する問題の一部分に光明を与えたに過ぎない。アプリケーション・プログラムにとっては，計算機の一次元的記憶装置内に，その各々の問題をいかにモデリングするかという広漠とした課題が残されている。インタラクティブ・コンピュータ・グラフィックスを，広範な分野に対して可能にするということは，問題―アプリケーション毎に任意の ― を，容易にモデリングさせうるということに等しい。…… こゝでいう"モデリング"とは，ある問題を表現する情報群を，データ・ストラクチュアに編成するとともに，これを操作するアルゴリズムの確立を含んでいる。

"問題"のモデリングを，個々のアプリケーション・プログラムにその全てを委ねるのではなく，汎用に利用可能なデータ・ストラクチュア・ファシリティをこしらえることが，インタラクティブ・コンピュータ・グラフィックスにとっての悲願である。多くの先駆的研究者達は，"問題"一般のデータをマッピングし得るデータ・ストラクチュアと，望みの操作を，易しく定義，指示しうる，高レベルのランゲージ・ファシリティの開発を試みてきた。

2.3.3 既存のデータ・ストラクチュアとその問題点

(1) データ・ストラクチュアに望まれる条件

"問題"一般をモデリングしうるデータ・ストラクチュアとは，どのような条件が課せられているかを吟味してみよう。それは，次のようにまとめられる。

① 構成要素が，各々任意数の属性を持つことができる。

② 構成要素間が，その内容（属性の値）の関連に基礎を置いて結び合わされ，検索ができる。

③ 構成要素，及びその属性間に，階層的構造を持たせることができる。

④ 構成要素に，ユニークなシンボルを対応させ，名前で参照できる。

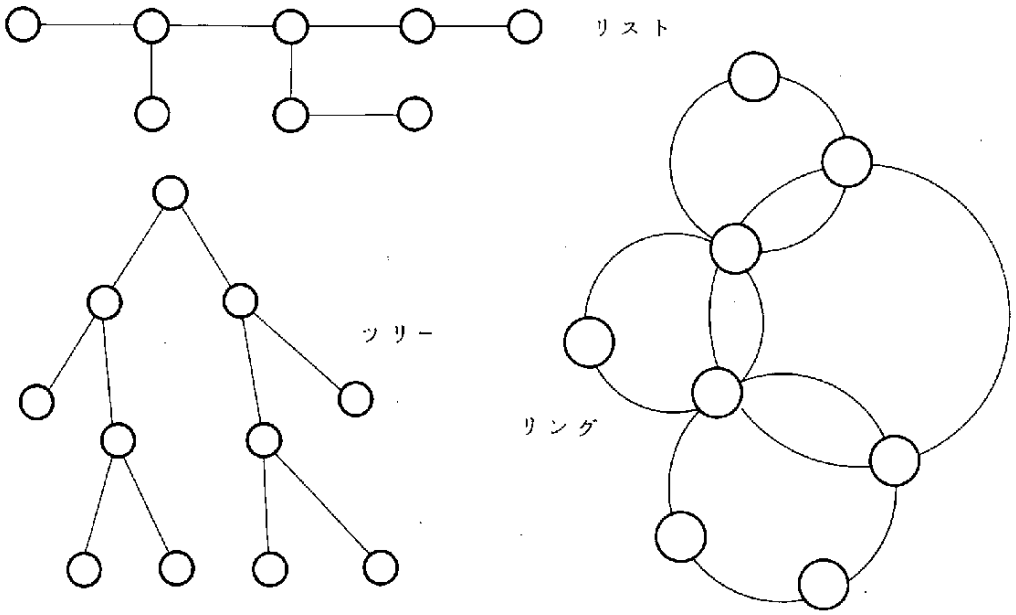
⑤ 構成要素の動的な再配置（変更，移動，追加，削除等）が可能。

⑥ 大量のデータが扱える。

(2) SKETCHPADの例

これらの懸案に，一応の解答を与えたのは，D. T. Plexの提出したPlesc Data Structure の概念である。これは，I. E. Sutherlandによって，SKETCHPADシステム（1962年M. I. T. Lincoln labo.）における，リング・ストラクチュアとして，実現された。リング・ストラクチュアは，リスト・ストラクチュアの特殊な変形とみることができ，

リスト中のメンバーを結びつけるポインターが巡環していて，リングの格好をしている。



リング・ストラクチュアは，次の機能的特徴を有している。

① リング上のどの位置からでも，そのリングの全メンバーの検索ができる。

② ポインターを修正するだけで，リング・メンバーの追加，削除，変更ができる。

③ マルチ・リングを採用することによって，多階層のデータ編成が可能。

④ 特定の処理アルゴリズム ― 処理ルーチン ― と結びついた，エレメントのタイプ・ワードが登録可能。

⑤ ガルベージ・コレクション等のテクニックによって，記憶スペースの効率的利用が可能。

I. E. Sutherlandが，SKETCHPAD システムで用いたリング・ストラクチュアは，n 成分のエレメントを単位として，多重のリングで結び合わせられるようになっている。リング・ポインターは，一つのエレメントに一つ以上含むことができ，あるリングのスタート・ポイントを含むエレメントを"hen"，他のエレメントを"chicken"と名づけ，階層づけがなされている。各エレメントは図形要素属性 ― たとえば，"線"とか"点"とかの ― を持ち，同属性の全エレメントは，"genelic block"と呼ばれるエレメントに向かって，リング結合されている。また，各genelic blockも，"supper genelic block" と呼ばれるエレメントに向って，リング結合されている。

このように，エレメント間を複雑に結び合わせる，多重構成のリング・ストラクチュアによって，SKETCHPADシステムは，2次元平面上の，直線と円弧からなる，任意図形の描写と操作を可能にした。

SKETCHPADによって開拓された2次元図形モデリングの手法は，その後1963年に，同じくLincoln labo. のT. E. Johnson による，SKETCHPADⅢシステムにお

いて，３次元図形モデリングへと，拡張的に引き継がれた。ＳＫＥＴＣＨＰＡＤⅢは，３次元立体のデータ入力，２次平面への平行投影，透視投影，及び回転操作等を含んでいる。

これらＳＫＥＴＣＨＰＡＤⅢシリーズの開発は，後続のインタラクティブ・コンピュータ・グラフィックスに，大きなインパクトを与えた。ＳＫＥＴＣＨＰＡＤそれ自体は，図形処理一般を対象としたわけではなく，ＣＡＤ（Computer Aided Design）のための，図形入力，変換処理を，マン・マシンの協業で行なわせる実験的アプローチであったが，対象となる"図形"が，ある範囲に限られたものではなかったし，D. T. Rossの示唆した，Plex Data Structure のもとに，これをモデリングする試みであったことから，"汎用データ・ストラクチュア"実現の夢を大きく促進したかにみえたためである。

汎用データ・ストラクチュアとして，リング・ストラクチュアが利用される場合，モデルの構成要素（エンティティ）は，エレメントと呼ばれる，論理的な記憶単位にまとめられる。— エレメントは，最小限，その表現するエンティティの属性を指標する識別子（タイプ・ワード）と，他のエンティティとの関連を表わすリング・ポインター（リンク）と，アプリケーション毎に任意の情報を含ませることのできる，データ部の，３つの部分から構成される。

リング・ストラクチュアをこしらえる上でやっかいな問題の一つは，この論理的な記憶単位としてのエレメントが，可変の大きさをとることにある。エレメントのデータ部に含まれる情報は，アプリケーション毎に（あるいは同一プログラム内でも場合によって）可変で，その質，量は予測し難い。ましてリンク部分は，リングに編成されたモデルの部分的変更によって，その必要量が動的に変化する。…… 従って，エレメントは，可変のｎ成分でなければならない。 — ①

リング・ストラクチュアにおける，第２の問題点は，非常に大量のデータを，ディスク，ドラム等の外部大記憶上に置く場合，複数ブロック（物理的レコード）にまたがるリング検索は，多数回の入出力アクセスを要し，処理およびレスポンスの速度を大きくそこなう恐れがあることである。— ②

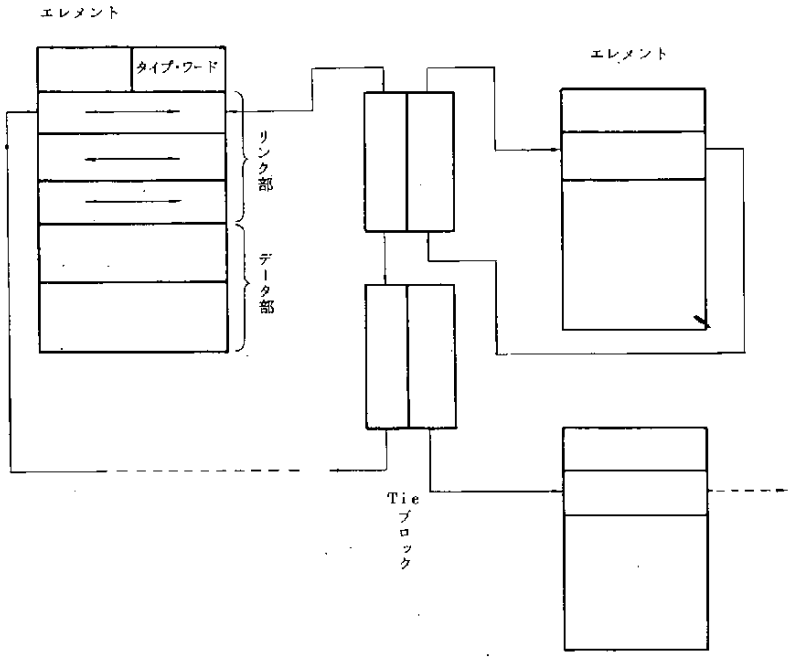
また，今一つの問題点は，全てのグラフィック・アプリケーションにとって，その"問題"のモデリングにおいて，リング・ストラクチュアが最適であるという保障がなにもない点である。ある特殊なアプリケーション分野は，通常のリング・ストラクチュアには到底マッピングし難い問題と，データを抱えているかもしれない。また，別のアプリケーション分野では，わざわざリング・ストラクチュアの複雑な処理を借りるには及ばないごく単純な問題を対象としているかもしれない。データ・ストラクチュアの汎用性と，個々のアプリケーションに対するオプティマイゼーションのかね合いの問題は，リング・ストラクチュアを採用する場合に限ったことではないが，最も難解なものの一つである。 — ③

以上，３項の問題を主な軸として，汎用データ・ストラクチュアを目ざす各種システムは，試行錯誤を重ねてきている。

(3) ＣＯＲＡＬの例

ＳＫＥＴＣＨＰＡＤシリーズに続いて，同じくM. I. TのW. R. Sutherland・らによって，

１９６３年に発表されたＣＯＲＡＬ（Class Oriented Ring Associative Language）の構想は，ｎ成分のエレメントと，Plex Structure の実現に新手法を用いた。ＣＯＲＡＬにおいては，エレメントは主記憶上の隣接した数ワードからなる固定長ブロックに対応され，リングの増加に伴って必要の生ずるリンクの追加は，Tie ブロックと云う，特殊なブロックを付加して，これをリングで結ぶ方法を取っている。



ＣＯＲＡＬでは，このような標準的なリング・ストラクチュアの編成，操作を実現する，各種ステートメントが提供されている。

(4) ＡＳＰの例

別のシステム…… たとえばＡＳＰ（Associative Structure Package）などでは，一つのエンティティは，隣接したワードからなるブロックにおさめられるのではなく，属性指標部，リンク部，データ部が，それぞれ独立のブロックを構成し，それらがポインターで結び合わされている。各ブロックは，その機能毎の識別子を持ち，追加されたブロックは，それぞれまたポインターで結びつけられる。これによって，個々のブロックは固定長でありながら，構成要素の"ｎ成分"を実現している。これに似た手法は，以降いくつかのシステムで採用されており，ダイナミックな変化をとげる構造体の表現と操作の任を，ある程度良く果たしている。が，反面，データ・ストラクチュアの構成法をますます複雑なものとした。特に，記憶装置の空き領域の管理に，複雑な処理を要するようになった。

(5) 記憶スペースの管理とアクセス

汎用のデータ・ストラクチュア・ファシリティは，これを利用するアプリケーション・プログラ

ムが要求するデータ総量を，あらかじめ予測することはできない。データ・ストラクチュアが充分に汎用であるためには，無制限ではないとしても，かなり大量のデータを収容し得るべきである。少なくとも，計算機の主記憶装置の物理量に制限されるようでは，問題にならない。すなわち，データ・ストラクチュア・ファシリティは，ディスク，ドラム等の第2次記憶装置上にデータ・ベースを置く必要がある。

第2次記憶上のデータは，適当な大きさのレコードに分割されねばならないが，新規に登録するエンティティを，どのレコード上に割りつけるべきか，あるいは，すでに登録済みのエンティティを，どのレコード中から参照すれば良いかを，いちいち明示的に指定させるのは，アプリケーション・プログラマにとって，大きな負担となろう。ユーザは，主記憶装置の状態や，第2次記憶上のレコード形式などを考慮することなく，あたかも無限の大きさの連続した記憶装置領域を占有しているかのように，その内のデータを参照できることが望ましい。このために，データ・ストラクチュア・ファシリティは，データ・ベース上と，主記憶上のデータ・トラフィック・コントロールをつかさどる，ファイル・マネージメントの機能を含むべきである。この機能の下では，ユーザのアプリケーション・プログラムにとって，自分の使用するデータ・ストラクチュアを収容した，第2次記憶上のファイルは，"透明"（transparent）である。ユーザ・プログラムにとって，第2次記憶装置は，主記憶装置の延長の"Virtual memory"として位置する。

この要求を満たすための一つのアプローチとして，いくつかのデータ・ストラクチュア・システムにメモリ・ページング・テクニックが導入された。ページング・エンバイロンメントの下では，データはある大きさの連続領域（ページ）に分割されて第2次記憶上に格納され，各ページはそれに属するデータが参照された時に主記憶装置内にローディングされる。主記憶装置は最低一ページ分の領域が割り当てられており，別のページが要求されて主記憶に余裕がなくなると，古いページが追放されてその領域に新ページがローディングされる。主記憶装置に余裕があれば複数のページが同時に存在し，また，参照が予測し得るときは，ページの先読みが行なわれる場合もある。参照されたデータが更新されると，そのページが主記憶装置から追放される時に，元の第2次記憶上にリライトされて，その最新情報が保たれる。

これらは，ＴＳＳでしばしば用いられる，ページ・スワッピングと本質的に等価である。ＴＳＳにおいては，データ領域に限らず，プログラムも含めた全領域がスワッピングの対象となるのが通常であるが，インタラクティブ・コンピュータ・グラフィックスの下では，特にデータ領域に対してこの処理が要求される。ページングへのアプローチには2種類あり，その一つはハードウエア及びＯＳによるページング（２次元アドレッシングを用いるような），いま一つは，プログラム―データ・ストラクチュアを提供するサービス・プログラム・システム―によるページングである。前者のエンバイロンメント下では，グラフィック・システムがページングに気をつかう必要はない。実際General Motors で用いられたＡＰＬシステム等は，ＩＢＭ／３６０－６７システムの，ダイナミック・リロケーション機能を利用している。

外部記憶と，主記憶間の自動ページ・スワッピングは，データ・ストラクチュア・ファシリティ

に，大量データへのアクセスを可能にする一方，いくつかの新たな問題を引き起した。第1に大記憶上の複数レコードにまたがるリスト検索速度の問題であり，第2にアプリケーション・プログラムのエラーもしくはシステム・ダウンに対するデータ・リカバリーと，ファイル保護の問題である。

第2の問題は，グラフィック・システムに限らず，ＴＳＳ等のシステム一般の問題である。

リスト検索の速度に関しては，アプリケーションがある問題をモデリングするのに，非常に大きなデータ・ストラクチュアを要する場合 ── いいかえれば，データが第2次記憶上に広く存在している時，ある項目に対するリスト検索は，数回の大記憶アクセスを要するかもしれず，インタラクティブ・コンピュータ・グラフィックスの必須条件である"即答性"をそこなう恐れが充分にある。

(6) Ｈａｓｈｉｎｇ のテクニック

この難問に対して，M.I.T のＦｅｌｄｍａｎらは，ｈａｓｈｉｎｇテクニックの応用を提唱した。ｈａｓｈｉｎｇは従来，プログラム・ランゲージ・プロセッサのシンボル・テーブル処理や，大きなデータ・ベースを使用するリアルタイムの商用データ・処理システムなどにしばしば用いられてきた手法である。ｈａｓｈｉｎｇ テクニックは，与えられたデータの一部をアクセス・ワードとして，これに疑似乱数発生の処理をほどこして得られた2進数字から，そのデータの大記憶上のレコード・アドレスを決定することに基礎を置いている。たとえば，アクセス・ワード，Ａ(Ｘ)を持つデータＸのレコード・アドレス，Ｌ(Ｘ)は，次式で与えられる。

Ｌ(Ｘ)＝Ｈ(Ａ(Ｘ))；

ここでＨはｈａｓｈｉｎｇ 関数であり，Ｘはチェック・ワードと呼ばれる。

Ｆｅｌｄｍａｎは情報の基本的構成をＡｔｔｒｉｂｕｔｅ，Ｏｂｊｅｃｔ，Ｖａｌｕｅの三者の関係(ｔｒｉｐｌｅ)に整理して表現した。こゝではある情報のまとまりA，$\overline{O}$，Ｖ(ｔｒｉｐｌｅ)を，その内のある要素を用いて指定する方法は，次の7つの場合がある。

1. Ａ($\overline{O}$)＝Ｖ
2. Ａ(？)＝Ｖ
3. Ａ(？)＝？
4. Ａ($\overline{O}$)＝？
5. ？(？)＝Ｖ
6. ？($\overline{O}$)＝Ｖ
7. ？($\overline{O}$)＝？

？は未知要素，もしくは特に指定されていない要素を表わす。この7形式はＳｉｍｐｌｅ Ａｓｓｏｃｉａｔｉｖｅ Ｆｏｒｍｓ(ＳＡＦ'Ｓ)とよばれる。

ｈａｓｈｉｎｇテクニックによって，上記のＴｒｉｐｌｅを大記憶上にマッピングする際，それらが大記憶上のスペースにどのような頻度で割りつけられるかゞ重要である。あるスペース内のデータ密度は，スペースの全体量，データ量，データの特性(アクセス・ワードのかたより)，ｈａｓｈｉｎｇ関数の選び方など，4つのファクタによって左右される。これはシステム・パフォーマンスに

大きな影響を与えるであろう。

hashingにおいては，異った複数のアクセス・ワードから変換されたデータ・アドレスが，同じ番地に衝突する場合がある。（これを"conflict"又は"collision"という）非常に広大なスペース上に，データをまばらにマッピングすれば，conflictは実際上さけられるが，これは不経済なので，大抵の場合はconflictを別の方法で解決している。これは，あるデータがすでに割りつけられた他のデータと同じアドレスに割りつけられる時，このデータを前のデータからのチェイニング・リストにして，別のアドレスに割りつける方法である。たゞし，このような方法をとった場合は，誤検索をさけるため，このデータをユニークに識別する識別子 — 大抵はチェック・ワード — を，hashing でアドレスづけられたデータの中に含ませる必要がある。次々と多重にconflict しているようなデータの検索は，数回のアクセスを要し時間がかゝる。理想的な方向は，データが，与えられた大記憶スペース上にまんべんなく，conflict を最少にしてマッピングされることであるが，内容間の関連を構造化するデータでは，たとえば自然言語によるシンボリックな名称を持つデータを割りつけるような場合が多く，このようなデータに対して，上の条件を良く満足するhashing のアルゴリズムを確立するのは，なかなか難かしい。

Feldmanの示したtriple（$A(\overline{O})=V$）においては，Vが一意に決まらない場合が生ずる。たとえば，

SON of FRED are TOM, BILL and ……；

$A(\overline{O})=(V_1, V_2, \cdots\cdots, V_n)$；

のような場合である。……これを，multiple－hitと呼ぶ。multiple－hit を含むデータの参照に対しては，該当する全項目を引き出せねばならない。このためには，multiple－hit した各項目もチェイニング・リストで結びつけられている必要がある。また，次の形式の参照に対する検索を可能にするために……

$A(?)=?$

Aなる共通要素に関係する全ての$\overline{O}$及びVが各々リングで結ばれている必要がある。

以上のようなhashing データは，7形式のSAFの内，3形式についての検索の場合は，良く任を果たす。3形式とは次のものである。

$A(\overline{O})=V$

$A(\overline{O})=?$

$A(?)=?$

他の形式のSAFに基づく検索要求は，きわめて煩雑な手順により，多数回の第二次記憶へのアクセスを経て満たされるであろう。

Rovnerはこの問題を，hash table を3重にこしらえることで解決を計った。

	Access−word	Check−word
A − space	A	$\overline{O}$
$\overline{O}$ − space	$\overline{O}$	V
V − space	V	A

これにより，与えられたＳＡＦの形式によって，３つのうち適したspaceから検索すれば，求める情報は最小の二次記憶へのアクセスで得られる。すなわち，A($\overline{O}$) = V，A($\overline{O}$) = ?，A(?) = ?の3形式はA spaceの検索から，また，?($\overline{O}$) = V，?($\overline{O}$) = ?は$\overline{O}$ spaceから，A(?) = V，?(?) = VはV spaceから……という具合に。しかしこれは，Triple の創成時，及び更新時には少なくとも3倍の手間と記憶スペースを要することを意味する。

hashing テクニックは既にいくつかのデータ・ストラクチュア・システムに取り入れられてきた。その代表的なものは，Feldman 及びRovner らによるＬＥＡＰ(ＭＩＴ Lincoln Labo. １９６８)や，T. E. JohnsonがＳＫＥＣＨＰＡＤ−Ⅲのversion−up に伴って開発したＲＰＬ(Relational Processing Language) などである。しかし，これらの実例も含めて，データ・ストラクチュア・システムへのhashing テクニックの導入はなお未解決の問題を多く含んでいる。それらは大きく次のように集約される。

1. Conflictの少ない，データの一様マッピング及び記憶スペースの最小化
2. n成分の(増減のあるデータの)網目になったデータの関連づけ(plex data structure)の実現
3. データ創成，更新時の効率向上

hashing テクニックは今後もさらに，現在における諸問題を解決すべく開発研究が試みられるであろう。なぜなら，hashing はインタラクティブ・コンピュータ・グラフィックスの核心である即応性のある情報検索を提供するから。…… 現在においても，たとえばデータの各構成要素にシンボリックな名称をつけ，そのメンバー名によるデータの検索を行なわしめるには，hashingテクニックの導入が最も効果的なアプローチであろう。

(7) 汎用データ・ストラクチュアへのアプローチ

データ・ストラクチュアの汎用性と，個々のアプリケーションの特殊性との間に横たわるジレンマについても，多くの解消の試みがなされてきた。

が，この問題に関する象徴的な一事実は，現在までに"汎用性"を標榜する様々なデータ・ストラクチュア・ファシリティが開発，発表されてきたが，そのいずれかを用いて得られた満足すべき成果 — 産業界の実用に充分耐えうる何らかのグラフィック・アプリケーションのような — についての報告は，かつてほとんどなかったということである。少なくも現在までのところ，大方の"汎用"データ・ストラクチュアは"実用向きでない"という皮肉は的を得た観測のようである。— もっとも，"汎用"のデータ・ストラクチュアを用いるにせよ用いないにせよ，"コンピュータ・グラフィックス"そのものがいまだ試行段階で，その成果を評価する程の実績に乏しい事情をかんがみれば，これを"汎用データ・ストラクチュア"への試みに対する最終審判とするのは酷に

すぎ，早計であろう。

何らかの結論づけを行う前に，データ・ストラクチュア・ファシリティの一つのゴールである"汎用性"そのものの意味について，もう少しつっこんで検討してみよう。

汎用性を目指すデータ・ストラクチュア・ファシリティには，2つのアプローチが考えられる。

その一つは，図形処理プログラムに必要な個々の特殊化された機能の全てを満足する，オール・マイティのストラクチュアを提供するやり方である。SKETCHPADやCORAL等のデータストラクチュアは，このアプローチへの試みであったといえよう。このアプローチは — もしそれが可能であるとすれば — グラフィック・アプリケーションにとって確かに強力な支柱となるであろう。…… いかなるアプリケーションも，単一のデータ・ストラクチュア・ファシリティのもとで，そのアルゴリズム，構造を意識することなく，任意の構造のデータを任意に操作することができるはずである。…… が，反面，そのようなデータ・ストラクチュア・ファシリティは，いかに単純なアプリケーションに対しても，オール・マイティであるが故に複雑に過ぎるストラクチュアを提供するであろう。また，アプリケーション毎に特殊化された問題は，ストラクチュア・ファシリティに次々とより高度で特殊化した機能を要求し，その全てをつくすストラクチュア・ファシリティは肥大化し，一方その効率は低下することが考えられる。しかも，個々のアプリケーションにとって必要な機能は，そのようにオール・マイティのストラクチュア・ファシリティの内，自分に関連のある一部分のみである。結局のところ，そのようなファシリティが実用になり得るであろうか？ — というより以前に，そのようなファシリティが果して存在し得るであろうか？ — この疑問に対する見通しの明るい解答は未だない。

もう一つのアプローチは，アプリケーション毎にユニークな，任意のデータ・ストラクチュアの編成を許すべく，サポートするようなファシリティのやり方である。Van Dam, David Evans らは，グラフィック・アプリケーション・プログラムは各々の目的に見合った"あつらえ"（tailored）のデータ・ストラクチュアを用いるべきであると主張した。アプリケーション・プログラムからの簡単な指示で，望みの構造を持ったデータ・ストラクチュアをしつらえるようなファシリティが可能であれば，グラフィック・アプリケーションの発展を妨げている環境はずっと改善されるに違いない。

このアプローチの最も顕著な例は，D. T. Rossを中心とするAED（ALGOL Extentioned for Design）システムや，EvansらによるDSPL（Data Structure Programming Language）の内に見ることができる。AEDでは，システムの提供する"フリー・ストレッジ"中にユーザがデータ・ストラクチュアを任意に構成できるようになっている。AEDを用いるプログラムにとって記憶スペースは主記憶装置の物理量に制約されず，2次記憶装置を含めた大容量を使用でき，そのある部分は必要に応じてシステムが主記憶と2次記憶間をスワッピングする。情報の記憶単位は，"bead"と名づけられ，bead は物理的に隣接した主記憶上のワードの並びである。AEDフリー・ストレッジ・パッケージは，ユーザからの要求に応じて任意個のbeadを割りつけ，また，それが返却された時の解放を管理する。bead の管

理には3つのモードがある。

1. 固定長のbeadを割りつけるSpecial mode
2. 数種類の標準サイズのbeadを割りつけるRegular mode
3. 可変長のbeadを割りつけるGarbage Collection mode

最もベーシックなレベルのAEDシステムにおいては，これらのモードの選択やbead の利用の方法 — ストラクチュア編成 — は，このシステムを利用するプログラムの側にゆだねられている。

AEDフリー・ストレージ・パッケージでは，"integrated package" と称してプログラムのモジュール化がはかられており，ベーシックなプログラム・パッケージを基盤として，一般的なリスト処理を提供するランゲージへの拡張がねらわれている。

AEDやDSPLシステムに代表される，"あつらえ" のデータ・ストラクチュアへのアプローチは，明らかにSKETCHPADシステムのシリーズのアプローチに比して，低いレベルに基礎を置いている。が，このような，ベーシック・ファンクションのプログラム・パッケージを徹底的にモジュール化し，より高いレベルのファシリティをその上に構築して行く方法は，システムの汎用性の観点からも，拡張可能性という観点からも，注目すべきアプローチであろう。グラフィック・アプリケーションの開発を志ざすユーザは，ベーシック・プログラム・パッケージを基礎として構築された種々のサブシステム — たとえばそれは，ある種のランゲージ・ファシリティを意味する — を利用することもできるし，また，ベーシック・プログラム・パッケージについて熟知し，直接にその機能を利用したり，必要に応じた拡張をほどこして，自由なストラクチュア操作を実現できる。

2.3.4 リング・ストラクチュアを用いた図形情報表現と操作の実際

リング・ストラクチュアを，汎用のデータ・ストラクチュアとして採用した各システムが，どのように問題をモデリングしているかについて，図形情報の場合に限って検討してみよう。

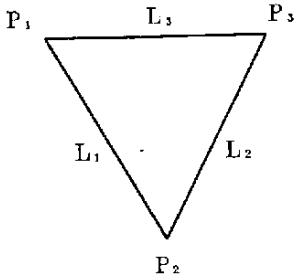
(1) リングの構成要素と結合

SKETCHPADシステムやCORALシステムにおいて用いられた図形情報ストラクチュアの概観は図Aのようなものである。

図でLINE L_1 と L_2 は共通の端点 P_2 を持っているので，これら3つのエレメント・ブロックを結ぶリングが設けられている。同様に L_2，L_3，P_3 ; L_3，L_1，P_1 の組も各々リングで結ばれる。L_1，L_2，L_3 はまたPICTUREのエレメント・ブロックとリングで結ばれる。

リング・ストラクチュアによって，同じ3角形を表現するのに，もう一つの方法が考えられる。図Bのストラクチュアでは，L_1 を現わすリングは，その両端点 P_1，P_2 を結ぶリングからなっている。同様にして，L_2，P_2，P_3 ; L_3，P_3，P_1 を結ぶリングが設けられている。PICTUREエレメントが，L_1，L_2，L_3 とリングで結ばれる点は前例のストラクチュアと同じである。

いずれのストラクチュアが図形表現に適しているかは一概に決め難いが，後者の構造は各エンテ



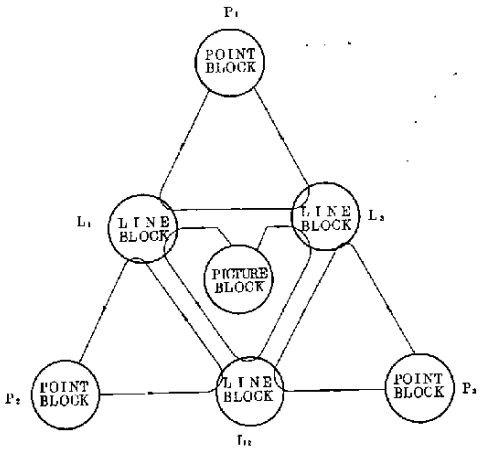


図　A

ィティ間の階層関係も良く表現するものと思われる。また，前者の構造は，ノード間を結ぶ枝が，何らかの資質と値を持つ時 — たとえば回路模型のような場合（図C参照） — に適した表示法と考えられる。

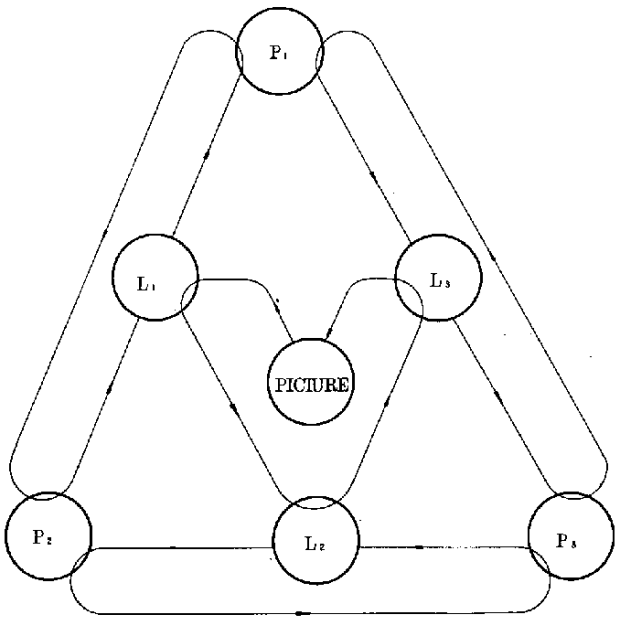


図　B

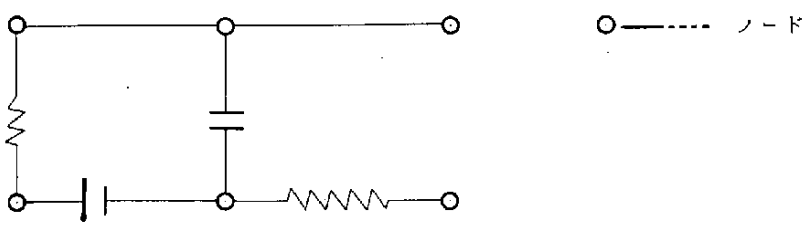


図　C

前者のリング構造を，J．C．Gray（University of Cambridge）がASP（Associative Structure Package）システムの記述に用いたノーテーションで表現すると，図Dのようになる。

同じく，後者のリング構造は，図E左のノーテーションで表現される。

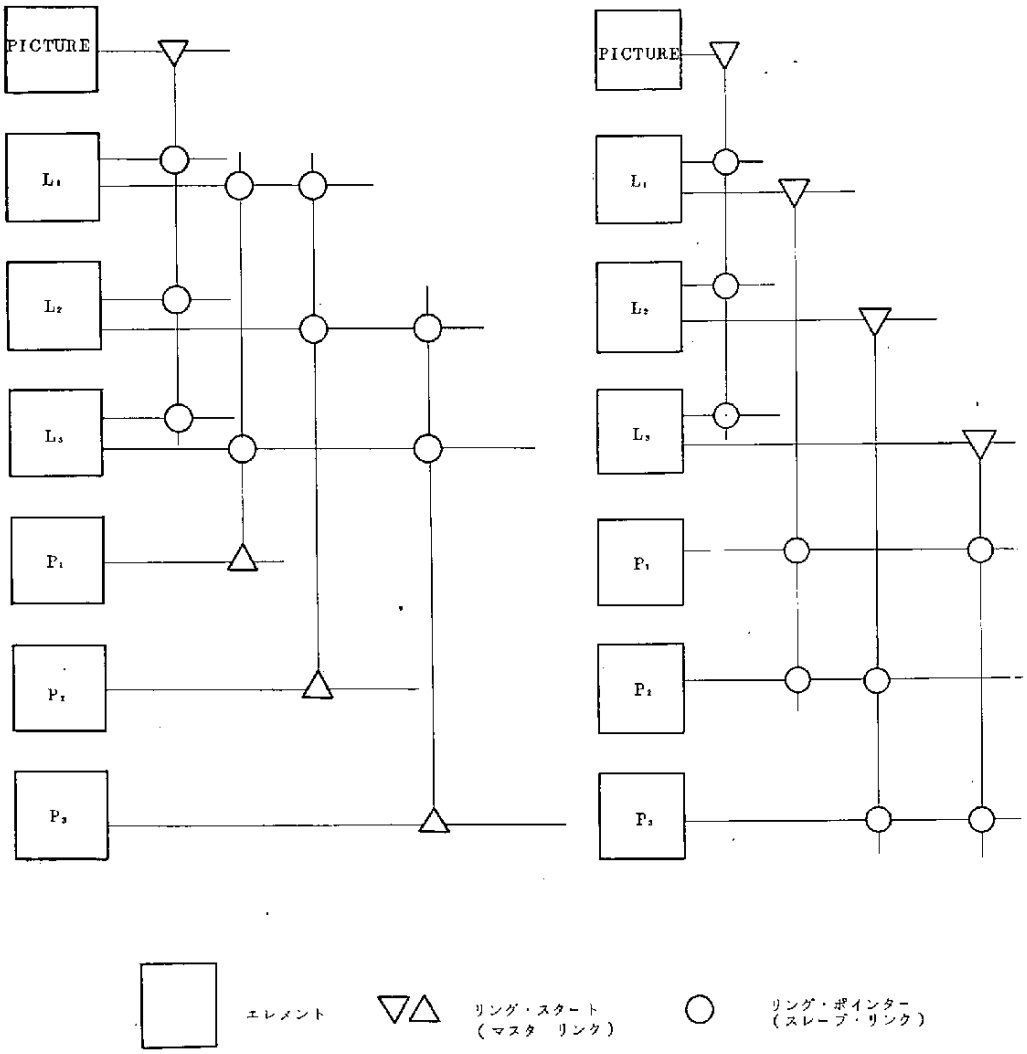


図 D　　図 E

(2) 階層関係の表現

ストラクチュアの構成要素間に階層関係を持たせることは，図形情報の記憶には特に重要な要件である。リング・ストラクチュアにおいては階層関係の表現はどのように実現されているであろうか。

右図で

A, B, C, ……エレメント

r_1, r_2, r_3, ……リング

r_{1a}, r_{2c}, r_{3b}, ……リング・メンバー（リンク）

とすると,

$r_1 \in \{ r_{1a}, r_{1b}, r_{1c} \}$

$r_2 \in \{ r_{2c}, r_{2d} \}$

$r_3 \in \{ r_{3b}, r_{3e}, r_{3f} \}$

$A \in \{ r_{1a} \}$

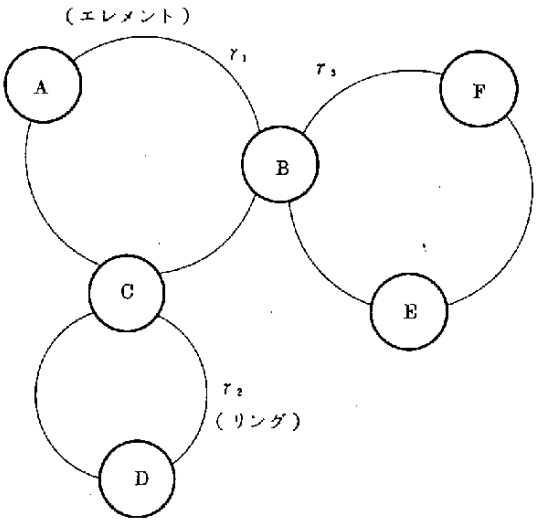
$B \in \{ r_{1b}, r_{3b} \}$

$C \in \{ r_{1c}, r_{2c} \}$

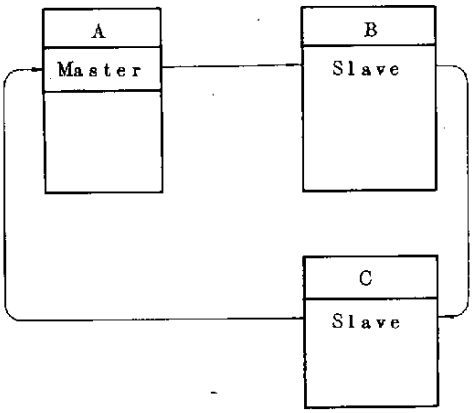
$D \in \{ r_{2d} \}$

$E \in \{ r_{3e} \}$

$F \in \{ r_{3f} \}$



こゝでリング・メンバー r_{1a}, r_{2c}, r_{3b} を他のリング・メンバーから区別して，"マスター・リンク"と名づけ，他のものを"スレーブ・リンク"と名づけて，その間に階層をつける。つまり各リングは必ず一つのマスター・リンクと任意個のスレーブ・リンクからなるものとする。



さて，ストラクチュア内の構成要素（エンティティ）— いいかえればエレメント — 間の階層づけを，このマスター・リンク，スレーブ・リンクの関係から規定することにする。…… すなわち，いまエレメントA, B, Cがあり，それらを結ぶリング r_1 のメンバー r_{1a}, r_{1b}, r_{1c} があるとする。

そして,

$r_{1a} \succ^{*} r_{1b}, r_{1c}$（$r_{1a}$ はマスター・リンク）

とした時,

A≻B, Cと規定する。

また,

A≻B； B≻Cならば，A≻Cと約束する。

---

* ≻の記号は，左項が右項を従える。…… すなわち，左項が右項のmasterであることを示す。

すると前掲の図では,

A≻B, C;

C≻D;

B≻E, F

また,

A≻D, E, F — となることがわかる。

このようにすると,一つの閉じたネットワークをなすリング・ストラクチュアに関しては,その全構成要素間に,任意のレベルの階層規定を与えることができる。たゞし,この階層は決して巡還するような — たとえば, A≻B, B≻Cでしかも C≻Aであるような — 構造になってはならない。そのような結果をもたらすリングをつけ加えてはならない。一つのネットワーク中の階層間に矛盾が生じないようにするのは,おそらくネットワークの編成者(それがユーザ・プログラムの場合であるか,システム・プログラムの場合であるかを問わず)の責任に帰さねばならないであろう。

こうした,階層構造の導入によって,ストラクチュア中のある要素に対するオペレーション — 消去,混合,変換 — が,他の要素に与える影響 — たとえば,ある線が消されると,それにつれてこの線上の全点が消されたり,逆にある点が消されると,この点を端点に持つ全ての線分も消去されるといった具合の — を自動的に決定するアルゴリズムの確立が許される。(SKETCH-PADシステムにおいて, I. E. Sutherlandはこのような機能を" Recursive function" と呼んでいる)

(3) リング・ストラクチュアと図形操作

リング・ストラクチュアにマッピングされた図形情報と,その変換処理(移動,回転,拡大縮少,シザリング,消去, etc)のアルゴリズムとの関連について考えてみる。

図形の線形変換には,同次座標系(homogeneous coordinates system)による,マトリクス演算が有効であることが良く知られている。リング・ストラクチュア上の図形情報に対して,この同時座標系を用いる変換や,図形の一部に対する消去要求を,データ・ストラクチュア上でどのような手順で実現するかは,システム・レスポンスの速度にかゝわる重要事項である。

○リングのハンドリング

いま,単純なモデル — 3角形 — について考えてみる。一つの3角形についてのデータのリング・ストラクチュアによる表現法は次図のようになる。図のようなデータを同時座標系による変換マトリクス演算をほどこすに当って,変換結果がディスプレイ画面上に現われるだけ — データ・ストラクチュアの側は元のまゝで変化を受けない — とすれば,ストラクチュアから,ディスプレイ・データを創り出す際に,3角形の各頂点に変換マトリクス演算をほどこせば良い。ストラクチュア内で,各頂点の座標データが,各々,

$P_1 = (x_1, y_1, 1)$;

$P_2 = (x_2, y_2, 1)$;

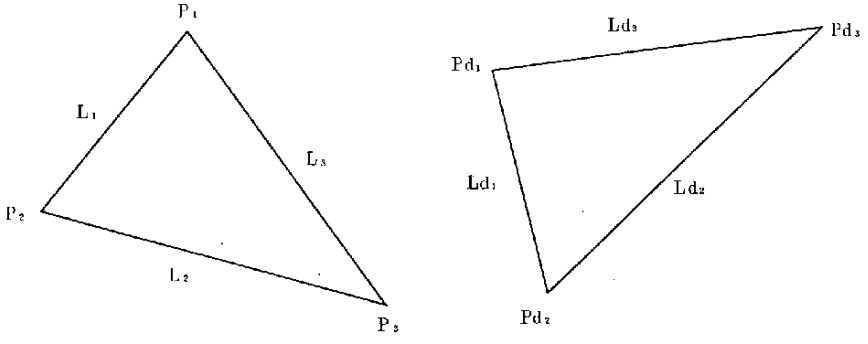
$P_3 = (x_3, y_3, 1)$;

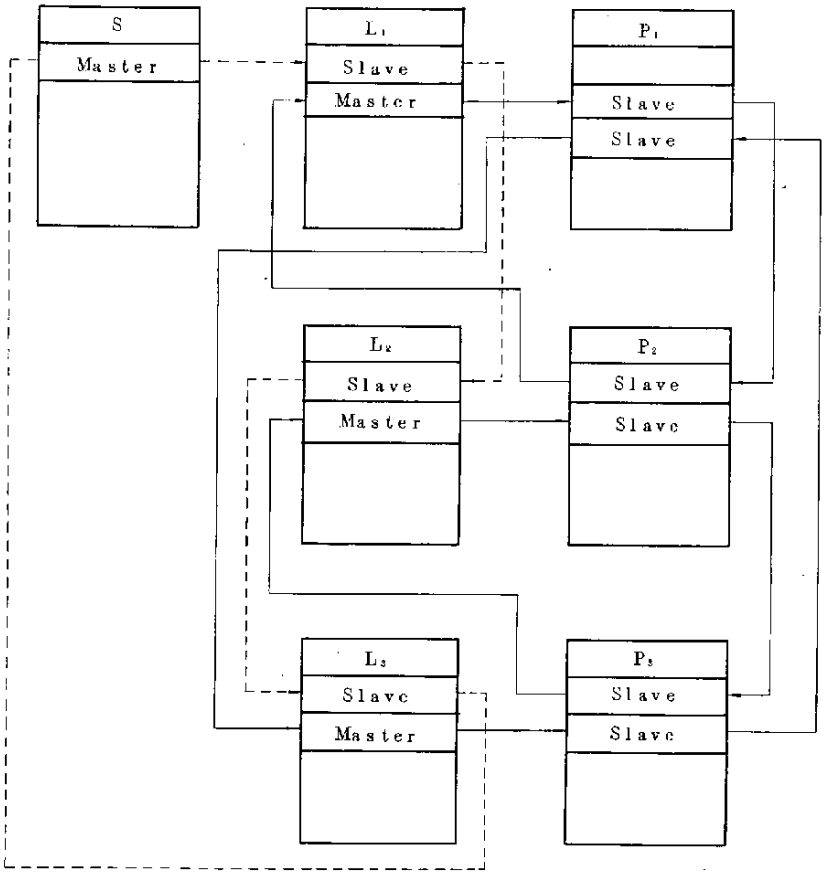
を示していたとすると，ディスプレイ画面上に現われるべき変換済みの各頂点の座標 $Pd_{1 \sim 3}$ は，

$Pd_1 = T \cdot P_1$　　（Tは変換マトリクス）

$Pd_2 = T \cdot P_2$

$Pd_3 = T \cdot P_3$　　で求められる。





3角形Sについての登録エレメントからリングをたぐり，L_1 ，L_2 ，L_3 の順に，その各端点についてこの演算をほどこしながら，ライン・ベクトルを創り出して行けばよい。

では，CRT画面上の図形のみでなく，それに対応したデータ・ストラクチュアそのものに，これらの変換結果をもたらしたい場合にはどうであろうか。これにはいさゝかやっかいな問題が存在する。上の例で，3角形についてのデータ・ストラクチュアに座標変換をほどこすには，P_1，P_2，P_3 をそれぞれ，Pd_1，Pd_2，Pd_3 で置き換える必要がある。しかし，これを先程と同じ手順で変換するとすれば，3角形の各頂点はL_1，L_2，L_3 の各頂点として2度づゝ変換を受けることになり，P_1，P_2，P_3 はそれぞれT・(T・P_1)，T・(T・P_2)，T・(T・P_3)と置きかえられることになってしまう。より複雑な図形に対するストラクチュア ― 網目の交錯している図形などでは特に ― では，事情はますます複雑になる。各リング・メンバーは一つの変換について，重複して更新を受けてはならない。

この問題への対処法の一つは，リングをたぐりながら，メンバーに変換をほどこして行く過程で，変換済みのメンバーをマークし，重複変換を防ぐことである。― しかし，この方法はメンバー・エレメント毎にマーク情報を要し，変換を終了した時に変換過程を再度トレースしてマークを消去する必要がある。

第2の方法は，すでに変換済みのメンバーをスタックし，このテーブルを参照しながら変換をほどこすべきメンバーを決定して行くやり方がある。― この方法は，テーブル領域と，参照のための多くのステップを要する。複雑な図形の場合には，この条件は致命的になろう。

第3には，エレメント毎に，その最も基礎的なメンバー・エントリー ― たとえば，座標(点など)を示すエレメント ― の全てを結ぶリングを設けておいて，変換はこのリング・メンバーに対してのみ行う ― 図例では3角形のエレメントSと点P_1，P_2，P_3 を結ぶリングを作っておく方法がある。しかし，あるエレメントに対して，どのエレメントが最も基礎的なメンバー・エントリーであるかを決定し，それらをたゞ一度だけ通るリングを創るプロセスのアルゴリズムは，この図形のデータ・ストラクチュアに対する，うまい変換をほどこすアルゴリズムと同意であり，問題は少しも解決しない。

第4には，変換結果を元のストラクチュアとは物理的に異るストラクチュアとして新設し，変換の終った過程で，元のストラクチュアを消去し新らしいものにかえる方法である。― この方法は，リング・ポインターのつなぎ変えなど，処理の複雑さは一番で，より多くの記憶領域を要する。

こゝにあげたいくつかの解決策はどれも，処理効率の面でも，アルゴリズムの簡潔さという面でも，あまりかんばしいものではない。より簡潔で効率の良い方策が開発さるべきであろう。

○図形の消去

リング・ストラクチュアに編成された図形の一部を消去する操作について考えてみる。

ある図形モデルが，階層的構造を有している場合，その"階層"は大きく2つの種類に分けて考えられる。

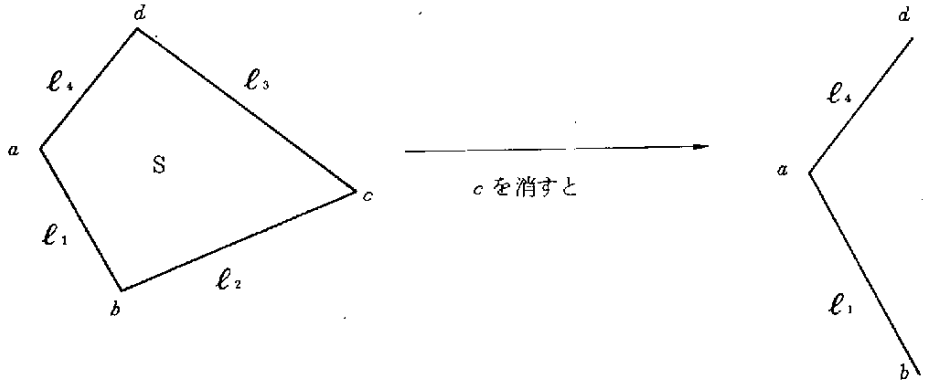
図形Aが，a_1，a_2，a_3，……a_n の要素を含む，すなわち，

$A \in \{ a_1, a_2, a_3, \cdots\cdots a_n \}$ として，

1. Aについて，その一要素 a_i を取り去れば，Aが存在し得なくなる場合 ― Aが幾何学図形の概念を含んでいる場合。

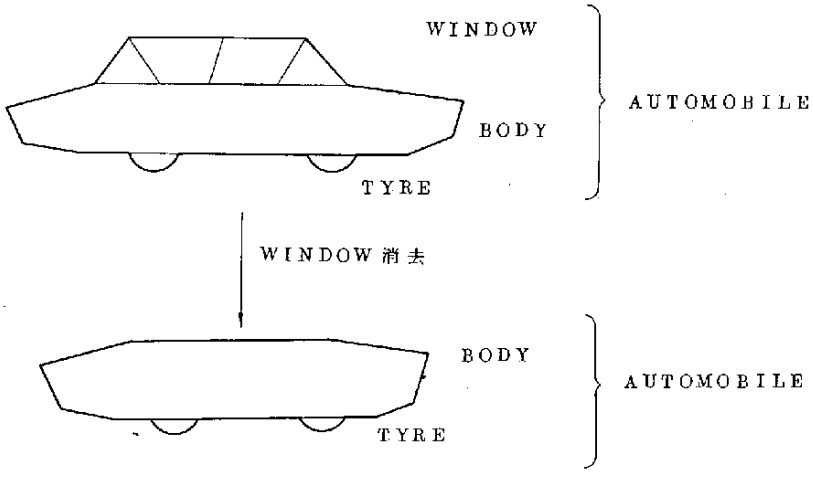
2. a_i が消えても，Aはその残りの要素の集合として残る場合 ― Aが無規定の集合の場合。

1については，たとえば多角形S（ a，b，c，d ）を考えるとき，その頂点が一つでも消えた時には，多角形Sはその条件を失う。（下図参照）



次に，2の範中の階層構造について考えてみる。図形モデルでも，1のような階層構造ではなく，2のような無規定集合としての階層構造を持たせたいという要求も生ずると思われる。

たとえば，自動車の車体設計のアプリケーションを想定してみよう。そこでは，車体の各部分―窓，ボディ，タイヤ等々 ― を順次描いて行き，それらを組み合わせたり，また気に入らない部分を消去，修正して行く操作が必要となろう。下図の例で，ＷＩＮＤＯＷの形がユーザの気に入らず，消去を指示したとする。この場合にＷＩＮＤＯＷを含む，より高い階層のエントリーであるＡＵＴＯＭＯＢＩＬＥが，1の例のように消去されてしまうのは不便であろう。こうした図形では，その一部が消されても，残りの部分を含めたエントリーが残る必要がある。

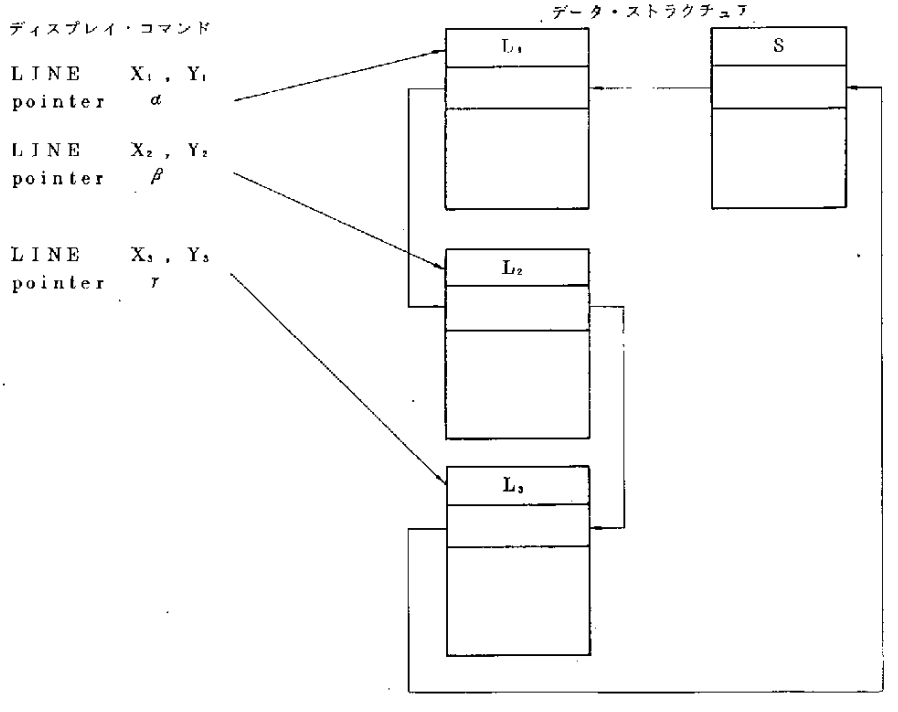


そこで，こうした操作を許すため，１とは異って，エレメント間の幾何学的図形としての概念を含まない結合ができるとよい。これはリング自身の内に持つ識別情報でも良いし（リングの一部に１，２のリングのタイプを区別する情報を持たせる），また，オペレーション・モードの区別でもよい。前者は，ストラクチュア内に，その構造の性質を固定するやり方で，後者は，それを使用者が任意時点で選択するやり方である。

○ライトペンによる図形ピッキング

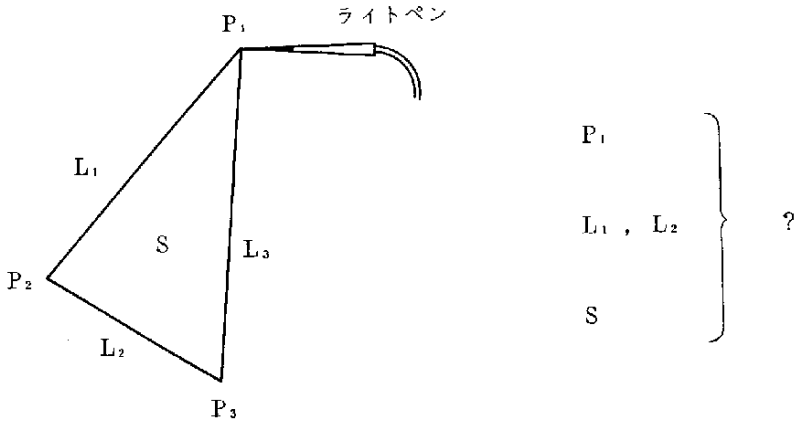
インタラクティブ・グラフィック・システムでは，現在ＣＲＴ画面上に表示されている図形の一部を，ライトペンによってピッキングし，その部分に何らかのオペレーションをほどこす機能が欠かせない。

これを実現するためにはライトペンがピッキングした図形の部分が，問題をモデリングしたデータ・ストラクチュア内のどの要素に相当するかの情報を得ることが必要である。ライトペン・ピッキングに対する金物的な反応は，ＣＰＵに対する割り込みとともに，その図形を表示したディスプレイ・コマンドのアドレスを返すのが通常のやり方であるが，グラフィック・インターフェイスのプログラムは，このコマンド・アドレスからその図形部分に対応するデータ・ストラクチュア内のエントリー・エレメントをたぐるアルゴリズムを含んでいる必要がある。このことは，ディスプレイ・コマンド列の随所に，データ・ストラクチュアの各エントリー・エレメントに対するポインターか，あるいはこれに代る情報の存在が必要とされるであろう。



しかし，このアプローチには明らかな限界がある。その一つは，図形の階層構造識別の困難性である。

ある図形の一部が，ライトペンでピッキングされた時，その図形中のどのレベルの階層が目標とされているかを，プログラムはどのようにして判別すれば良いであろうか？



上図のようなピッキングは，果たして，点P_1 を指したものであろうか，L_1，それともL_2，あるいはまた3角形S全体を目標としたものであろうか？…… これがピッキングによる割り込み情報だけでは解釈し難いことは明白である。オペレーションのシンプルさをいさゝか犠牲にして，そのような階層指示にライトペン・ピッキング以外の附加情報をつけ加える — たとえば，階層レベル毎に異ったファンクション・キーを対応させて，ピッキングと一緒にその一つを押す — のも，一つの対処法である。しかし，これも全ての問題を解決する訳ではない。図形がいわゆるメッシュ構造をなしている場合には，その階層は複雑である。前図の例でも，ファンクション・キーで線の捕捉を指示したとしても，その線とは，L_1 であろうか，L_2 であろうか？…… 点P_1 は，両線分に属しているのである！

より複雑なメッシュ構造の図形においては，ある要素が，どれだけ多くの他の要素に従属し，また，何レベルの階層を持つストラクチュアの，第何番目のレベルの階層に属しているかについては，全く計り難いといえよう。そのような図形において，表示図形の部分ピッキングを，図形の任意の階層に対応づけようという試みは，実に気のめいる難問である。

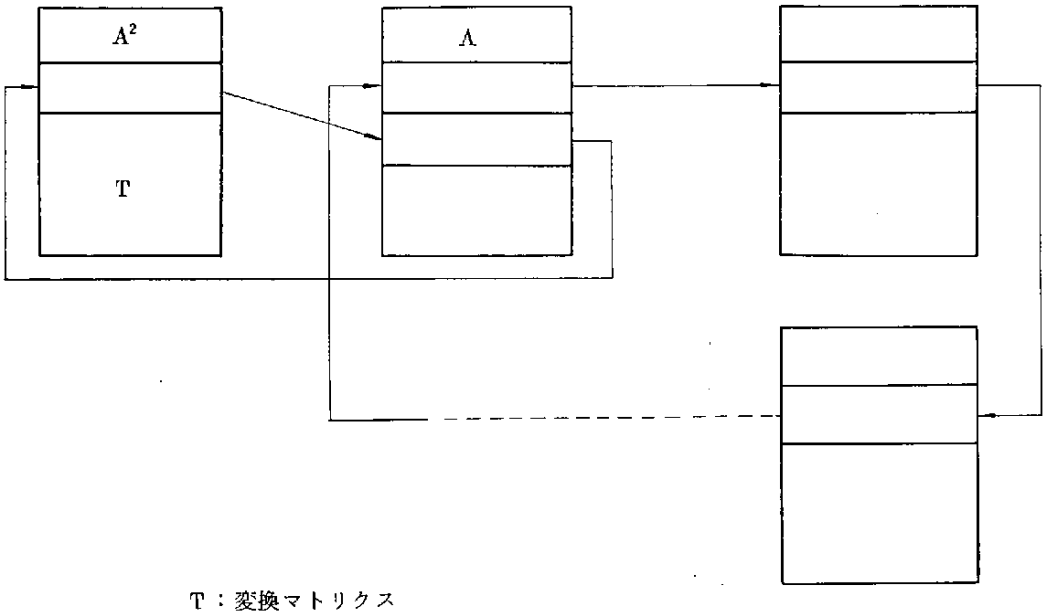
この問題をスマートに解決できたシステムは，今までのところ皆無である。"インタラクティブ"の真髄ともいうべき，ライトペン・ピッキングが，表示図形捕捉に充分生かし得ないでいるのが現状である。表示図形ピッキングを，階層捕捉と結びつけるアルゴリズムを見つけ出すことは，これからの課題となろう。

○Subpicture 及び repetition

データ・ストラクチュア上に，既に登録済みの図形の一部，もしくは全体を，他の図形の部分，もしくは全体と，いろいろに組み合わせて，新らしい図形モデルを構成させる機能があると，合同図形や，相似図形をしばしば用いる場合に便利である。

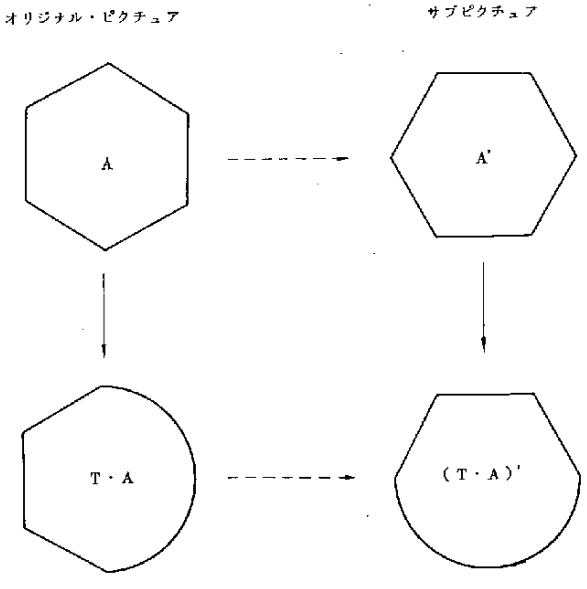
図形Aに線形変換をほどこした図形A'が，常に，原図形Aのストラクチュアとは別のストラクチュアとして編成されるのでは，処理時間，記憶領域ともに不経済であろう。A'を，その原図形，

Aのストラクチュアに依存したストラクチュアとして登録できると，ある種の目的 ― 連続幾何模様の表示のような目的 ― には，有効である。その一つの表現法は，次の図のようになる。こうした表現をとられるA'は，Subpictureと呼ばれる。



T：変換マトリクス

サブピクチュアは，それ自身では，ストラクチュア上の部分構成を持っていないので，部分に対する操作 ― 消去，追加，移動，回転等 ― は許されないであろう。また，サブピクチュアの原図形（オリジナル・ピクチュア）の変換操作は，サブピクチュアそのもののパターンにも，変化をもたらす。（次図参照）



あるサブピクチュアのオリジナルがまた，その他のサブピクチュアを含んでいてもかまわない。…… ただし，その関係が巡還するものは，パターンが定まらないために不可となる。このことはまた，リングの階層が巡還していて，矛盾を持つ場合を意味する。

このような，サブピクチュアの導入は，連続模様を描いたり，同じ形状を持つ部品図を各種に組み合わせる ― 回路図のように ― アプリケーション・プログラムからは，充分に珍重されるであろう。これらの機

能は，すでにＳＫＥＴＣＨＰＡＤシステムにおいて，ほぼ完全に実現されている。

以上に述べてきた，汎用データ・ストラクチュアへの各種アプローチは，いずれもまだプロトタイプのレベルに留まっているといわざるを得ない。これらのストラクチュアが，現実の各種アプリケーションにとって，有効強力であるという，プラクティカルな裏づけが乏しいからである。また，今後も続けられるであろう汎用データ・ストラクチュアの研究開発が，やがては報いられるに違いないという保証も充分ではないのが，残念ながら現実である。一方には，このような試みを，すでにあきらめているか，もしくは，"無駄な努力"として軽視している研究者も少なくないと思われる。……しかし，そうした断定さえも，充分に論理的な根拠 ― 汎用データ・ストラクチュアの，現実の"必要性"をもくつがえす程の ― に基礎を置いているようにも思えない。現段階では，早急な結論づけよりも，より多くのプラクティカルな試行が蓄積されるべきである。

2.4　ランゲージ・ファシリティ

2.4.1　グラフィック・ランゲージ・ファシリティ

グラフィック・CRT・コンソールを前にした，ユーザをサポートすべきグラフィック・システムの機能について，これまでに2つの条件を与えてきた。その一つは，"インタラクティブ・ジョブ・コントロール"であり，いま一つは，"汎用データ・ストラクチュア"である。インタラクティブ・グラフィック・システムを，充分に"ユーザ・オリエンティッド"にこしらえるためには，これら2つの条件の上に，さらにもう一つの条件，…… 使いよく，強力にこしらえられた，"グラフィック・ランゲージ・ファシリティ"の提供が課せられる。

"グラフィック・ランゲージ"なるものに必要とされるであろう諸機能を，次に列挙してみよう。

① グラフィカル$\overline{\text{I}}/\overline{\text{O}}$・インターフェイス

② 割り込み制御

③ ストラクチュア操作

④ 複数プロセッサ・システム（host computer と，satellite computer）の下では，プロセッサ間の結合

それに，グラフィック・ランゲージに限らず，ランゲージ・ファシリティ一般に課せられる条件であろうが，……

① 使い易さ

② 汎用性

③ 効　率

④ マシン間の互換性

⑤ 拡張性

…… 等を満たすことが要求されよう。

これまでに開発されてきた，グラフィック・ランゲージに関して，便宜的な類別を試みると，次のようになる。

① ランゲージのレベルに関して

- プロセデュラル（procedural）
- プロブレム・オリエンティッド（problem oriented）

② 記述方式に関して

- written form
- pointing form

③ 割り込み制御に関して

- 同期処理

- 非同期処理

④ データ・ストラクチュアに関して

- ストラクチュア・セットを含む
- ストラクチュア・セットを含まない

2.4.2 プロセデュラル・ランゲージ

現在最も多く開発され，また実用に供せられているグラフィック・ランゲージは，FORTRANタイプのものである。その多くは，グラフィック・サブルーチン・パッケージの形を取っており，従来のFORTRANによるプログラムとのインターフェイスが考慮されている。プログラマが，グラフィック・CRTを通じた入出力を要する場所で，CALLステートメントを用いて，グラフィック・サブルーチンを明示的に組み込むようになっている。例を次に掲げる。

CALL LINE (x_1, y_1, x_2, y_2, n) ①

CALL CIRCLE (x_1, y_1, r, n) ②

CALL DISPLY ③

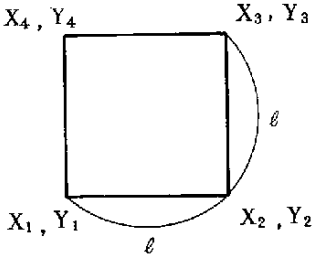
①は，CRT上の座標(x_1, y_1)から，(x_2, y_2)まで，輝度nでベクトルを表示するようなディスプレイ・コマンドの列を作り出すことを要求している。同じく②では，座標(x_1, y_1)を中心，rを半径とする円を，輝度nで表示するコマンド列を作り出すことを要求している。①，②で作り出されたディスプレイ・コマンドは，③の要求で，実際にCRTに表示されるであろう。

このような，FORTRANタイプのグラフィック・ランゲージは，従来のFORTRANプログラミングに要する以上の知識を必要としないという点でも，また，既成のプログラムにグラフィック・サブルーチン・コールを付加するだけで，CRT入出力をアプリケーション・プログラムに許すという点でも，便利なファシリティである。

FORTRANを基盤としたグラフィック・ランゲージのもう一つのタイプは，FORTRANコンパイラのプリ・プロセッサの形をとっている。こゝでは，上述のようなディスプレイI／$\overline{O}$をつかさどるサブルーチン・コールが，FORTRANに追加されたステートメントによって展開される。単純な図形 — たとえば正方形のような — は，ベクトル単位に分解した記述を与えなくとも，一つのステートメントで容易に表示される。前者のタイプのFORTRANプログラムが，下の①のようであるとすれば，これが後者のタイプでは，②のようになるであろう。

①	②
	DISPLAY RECTAN(X_1, Y_1, ℓ, 4)
CALL LINE	(X_1, Y_1, X_2, Y_2, 4)
CALL LINE	(X_2, Y_2, X_3, Y_3, 4)
CALL LINE	(X_3, Y_3, X_4, Y_4, 4)
CALL LINE	(X_4, Y_4, X_1, Y_1, 4)
CALL DISPLY	

このタイプのランゲージは，前述の通り，ＦＯＲＴＲＡＮのプリ・プロセッサ方式を採用しており，ＣＲＴ入出力に関する追加ステートメントを，適当ないくつかのサブルーチン・コーリング・シーケンスに展開して，ＦＯＲＴＲＡＮコンパイラに渡す働きをしている。このタイプのランゲージによって，ＦＯＲＴＲＡＮを用いたグラフィック・アプリケーションが，ＣＡＬＬステートメントと，アーギュメント・リストの羅列となってしまう。前者の欠陥はいく分カバーされ，プログラミングもさらに容易になるといえる。



ＦＯＲＴＲＡＮタイプのグラフィック・ランゲージは，いずれの場合にせよ，ストラクチュア操作を含まないのが普通である。また，アプリケーションに課せられている問題そのものを記述する＂Ｐｒｏｂｌｅｍ ｏｒｉｅｎｔｅｄ＂ランゲージではなく，問題を処理する手続きを記述する＂Ｐｒｏｃｅｄｕｒａｌ＂なランゲージである。従って，表示図形情報の動的な操作，変換に基礎を置いた，データ・ストラクチュアをこしらえること，及び，それらを操作することは，全て各ＦＯＲＴＲＡＮプログラマに個別に課せられていることになる。ＦＯＲＴＲＡＮプログラムでこれを実現するとして，そのプログラミング労力がどれ程になるかは，想像に難くない。この種のランゲージは，比較的単純な図形入出力を含むアプリケーション向きといえよう。また，ち密なマン・マシン・インタラクションをねらうアプリケーションにも，あまり向いていないといえそうである。

2.4.3　プロブレム・オリエンテッド・ランゲージ

より高レベルのランゲージ・ファシリティは，データ・ストラクチュア・セットと，その操作の機能を提供している。しかし，そのアプローチは実にさまざまである。ランゲージによって，その含むストラクチュアは，ＣＯＲＡＬのごとく，標準的なリング・ストラクチュアを提供するもの，ＡＰＬ（Ａｓｓｏｃｉａｔｉｖｅ Ｐｒｏｇｒａｍｉｎｇ Ｌａｎｇｕａｇｅ：前出）のごとく，ｈａｓｈｉｎｇによるデータ・マッピングと検索を含むもの等，種々のタイプが存在する。…… データ・ストラクチュアの種々の型については，前章で触れて来た。

データ・ストラクチュア・セットを含むグラフィック・ランゲージにおいては，構造をなしたデータを，容易に扱えねばならない。その条件は……

① 構造体のシンボリックな扱い

② 処理の進行に従って，動的に変化する構造体の操作

③ 速いレスポンスを与えるに充分な処理効率

これらの問題は，全て前章で論じてきたことであるので，こゝで改めて取りあげる必要はないが，ランゲージの側面からは，このストラクチュアの定義と操作を，何らかのステートメントと，どう対応させるかゞ問題となる。

いわゆる＂構造体＂の操作は，既存のランゲージ ― ＣＯＢＯＬ，ＰＬ／Ｉ ― でも，すでに取

り入れられている。こゝで論じている"構造体"が，それらと区別される条件は，

① 構造がダイナミックに変化する。

② 構造は，その物理的な位置関係によるのではなく，内容間の意味的関連による。

…… このような構造体を扱うランゲージに，ＬＩＳＰなどの，リスト処理言語があることはよく知られているが，グラフィック・ランゲージにおけるストラクチュア操作は，これらのランゲージにおける，構造体処理と似ている。

動的なストラクチュア・データの操作能力は，そのグラフィック・ランゲージが"Problem Oriented"であることの一つの条件をなしている。しかし，より基本的な条件は，ランゲージが，グラフィック・ＣＲＴ，表示図形，及びオペレータの応答と，プログラムとの関係を，どう位置づけているかにある。すなわち，ＦＯＲＴＲＡＮタイプに代表される，"Procedural"ランゲージでは，グラフィック・ＣＲＴはあくまで，プログラムの"入出力媒体"に留まっている。表示される図形は，プログラムの内部処理結果の出力であって，プログラムの進行過程そのものを表現する訳ではない。この場合，ＣＲＴへの図形表示は，形態こそ異なれ，プログラム上はＸ－Ｙプロッタ等の出力装置への出力と，本質的な差異はないのである。ＣＲＴのオペレータからのプログラムに対する応答にもまた，"Procedural"と"Problem Oriented"の差は現われる。"Procedural"ランゲージでは，ライトペン，ファンクションキー，etc の割り込み情報は，"一風変った"入力装置からの入力データでしかない。

では，"Problem Oriented"と呼ぶにふさわしいランゲージでは，ＣＲＴ入出力を，プログラム上にどう位置づけているであろうか。

Ｐ．Ｇ．Comba（Data Processing Division Scientific Center，（Ｉ．Ｂ．Ｍ．）New York）のくわだてた，ＰＬ／Ｉのグラフィック・ランゲージへの拡張においては，ＣＲＴに表示される図形は，プログラム内でシンボリックに参照されるグラフィック変数，および，構造体そのものである。たとえば，

ＣＬＡＳＳ１．ＣＬＡＳＳ２．ＮＡＭＥ＝ＬＩＮＥ（ＳＴＡＲＴＰＴ，ＥＮＤＰＴ）；

このステートメントでは，ＣＬＡＳＳ１，ＣＬＡＳＳ２，ＮＡＭＥ，ＳＴＡＲＴＰＴ，ＥＮＤＰＴは，それぞれ宣言済みのグラフィック変数で，これらの間をアサインメント・ステートメントを用いて，新たに関係づけている。このステートメントが実行されると，ＳＴＡＲＴＰＴ，ＥＮＤＰＴというグラフィック変数の持つ値 ― ＣＲＴに表示されている，２点のＸ，Ｙ座標値 ― を，ＬＩＮＥというファンクションで結合したもの ― ２点を結ぶ線分 ― に，階層を持ったidentifire―ＣＬＡＳＳ１．ＣＬＡＳＳ２．ＮＡＭＥ ― を対応させる。

また，次の例では，

```
FOR EACH LINE SUCH THAT POINTA IS END OF
        LINE OR POINTB IS START OF LINE DO ;
        Statement 1   ;
        Statement 2   ;
            :
            :
        Statement n   ;
END ;
```

プログラムが定義した，全ての線分属性を持つグラフィック変数の内，ＰＯＩＮＴＡの名称を持つ点属性変数を終端点とするか，ＰＯＩＮＴＢの名称を持つ点属性変数を始端点とするもの全てについて，ｓｔａｔｅｍｅｎｔ１～ｎの処理をほどこすことを意味している。

Combaの企てたＧＬ（Ｇｒａｐｈｉｃ Ｌａｎｇｕａｇｅ）では，３次元空間内での，物体の配置問題を解析しうる，強力なストラクチュア操作のステートメント群をも含んでいる。プログラムで作り上げた，問題のモデルは，なんらかの解析が加えられた後に，ＤＩＳＰＬＡＹのステートメントにより，その図形的形象操作 — ３次元物体の２次元投影，陰影等 — が加えられて，ＣＲＴに表示される。

Ｃ．Ｉ．Johnsonは，＂Ｐｒｏｂｌｅｍ ｏｒｉｅｎｔｅｄ＂な図形処理言語による，イメージのＣＲＴ表示には，２つのモードの実現が考慮されるべきであるとしている。…… その一つは，プログラムによる，ストラクチュア化されたデータの解析，操作の後に，その選ばれた一部が明示的な指定によって — つまりＤＩＳＰＬＡＹステートメント等に従って — ＣＲＴに表示されるモードである。このモードでは，＂Ｐｒｏｂｌｅｍ ｏｒｉｅｎｔｅｄ＂と，＂Ｐｒｏｃｅｄｕｒａｌ＂なランゲージの間に差はないといえる。ＧＬもなお，このレベルに留まっている。もう一つのモードは，プログラムによるデータ・ストラクチュア，及びグラフィック変数の操作の側面効果として，その図形的変化が刻々とＣＲＴ上に表示される，＂デフォルト・モード＂もしくは，＂ｍｏｖｅ＂ モードである。こゝでは，ＣＲＴは単なる図形入出力装置としてではなく，プログラムの進行状態，処理内容を覗く，＂窓＂として位置している。

プロブレム・オリエンティッド・ランゲージでは，また，ＣＲＴのオペレータが，ライトペンで表示図形に対して，あるいは，ファンクション・キーなどによって起すであろう，アクションは，単なるデータ入力としてではなく，プログラム処理の論理的構造の一端をなすであろう。すなわち，論理的な意味でのプログラム — 問題処理のアルゴリズム — は，もはや機械系の内に封じ込められた存在ではなく，マン・マシン・インタラクションの内に，ｏｐｅｎ－ｅｎｄｅｄに存在することになる。

こうした，＂Ｐｒｏｂｌｅｍ ｏｒｉｅｎｔｅｄ＂ のグラフィック・ランゲージ・ファシリティを実際にこしらえるには，多くの困難が克服されねばならない。そのファシリティは，充分に汎用のデータ・ストラクチュア・セットと，その操作を容易にする強力なステートメント群 — 今まで，どのランゲージに現われたよりも強力な — が内包されなければならない。他方，データ・ストラクチュアの汎用性，ステートメントの強力さが，現実の処理装置系と，処理速度等の点でうまく折り合えるであろ

うか？…… 現段階に関する限り，必ずしも明るい見通しばかりではない。しかし，このようなアプローチは現実に始められているし，いくつかの実験的成果が見い出されつゝある。Combaの提唱したGL（Graphic Language）の構想，IBMによるPGS（PLAN Graphic Support）などは，"Problem oriented"ランゲージへの，かなり大胆な企てを含んでいる。APLや，LEAPもまた，一部に，その試みが取り入れられている。

2.4.4 同期割込処理と非同期割込処理

グラフィック・CRTを用いる，図形処理プログラムの大きな利点の一つは，グラフィック・CRTに向かったオペレータ，もしくはプログラムのユーザと，CRTの付属入力用装置 — ライトペン，ファンクションキー，etc — によって，プログラムを"会話モード"で実行させられることにある。"会話モード"とは，プログラムとオペレータの両者が，常に"Read and Reply"の関係に立って，相手の出方を見守りながら，次の動作を決定して行くやり方といえよう。

しかし，CRTを介した，"マン・マシン・インタラクション"とは，果してこの"会話モード"だけで満足されるものであろうか。

たとえば，次の様な場合を想定してみよう。…… あるプログラム — まだデバッグ中であるかもしれない — が，実行中に，オペレータの期待にそぐわない誤動作を繰り返して，CRTへの間断ない出力命令を出してしまった，あるいは，オペレータからの指示を受けつけない（入力命令を出さない）状態のまゝループに入ってしまった。……

このような場合，プログラムの進行を停止，もしくは矯正するための処理は，CRTのオペレータの手に余ることになってしまう。

また，別な場合，…… たとえば，ムービング・アニメーションを行う図形処理プログラムの進行中に，オペレータが望んだ適当な時点で，処理を一担停止し，その時点での表示図形のハードコピー（X－Yプロッタ等への）や，又は，別の処理へのトラフィック・コントロールを要求したい時，このようなシステムでは，どう処理すれば良いであろうか。

以上の問題は，"Read and Reply"のシステム — いい換えれば，同期処理系では，本質的に解決しない。

少々回り道をしたが，前述したことは，グラフィック・プログラムに，CRTオペレータからの非同期割り込み処理を含ませるか否か，グラフィック・ランゲージ・ファシリティに，この機能のサポートを含ませるか否かの問題となって帰ってくるのである。

通常，CRTオペレータからの入力 — ライトペン，ファンクションキー，etc — は，プログラムへの非同期割り込みを伴って果たされている。しかし，この非同期割り込みを処理できるか否かは，グラフィック・プログラムの構造のいかんに依存しているといえる。すなわち，仮に，あるグラフィック・プログラムがなんらかの処理途中で，非同期に割り込まれ，この割り込み（それは，オペレータからのなんらかの指示であろう）に対応した処理をたゞちに開始したとする。…… もし，この割り込み処理過程で，このプログラムが割り込まれた時点に使用していたプログラム・リソース

（ルーチン）が共用されるようなことがあると，このプログラムがある特殊な構造を持っていない限り，割り込み発生前のプログラム・ステータスが破壊され，これを元に戻すことができなくなってしまう。このような処理を，支障なく行なえるのは，リエントラント構造のプログラムだけである。リエントラント構造のプログラムは，プログラムがあるルーチン通過中に割り込まれて，その割り込み処理中で同一ルーチンを通過したとしても，この割り込みを解いた時点 ― すなわち，元のルーチンの処理を継続した時 ― にも，その結果はいささかもそこなわれることはない。

ところで，既存のプログラム・ランゲージの大部分は，そのオブジェクト・プログラムをリエントラント構造にするようにはできていない。一般のＦＯＲＴＲＡＮプログラムが，リエントラントでないのは勿論のこと，ＣＯＢＯＬに至っては，一部のステートメント処理では，実行中に命令の一部を書き変える手法が良く用いられている。これらのランゲージで，完全な非同期割り込み処理を果たすプログラムを組むことはできない。こうしたランゲージによるプログラムの，ＣＲＴを介した，対オペレータとのインタラクションは，"Write and Wait ― Read and Reply" のサイクルのくり返し，― すなわち，ある一連の処理を果たし，その出力が終った時点で，プログラムの進行を停止して，オペレータの応答を待つ ― という形態か，あるいは，処理中に発生した，オペレータからの割り込みは，発生時点でただちにプログラム上で処理されずに，割り込み発生を指標するスウィッチがセットされるのみで，後に，プログラムが明示的な条件ステートメントでそれを問い合わせるまで，処理が遅延される方式かの，どちらかになる。

プログラマが，アセンブラ・ランゲージなどで，意識的にリエントラント構造をめざして組んだプログラム，または，オブジェクトをリエントラント構造に編集する機能を持ったランゲージで組んだプログラムにおいては，このような制約を受けることはない。リエントラント構造のプログラムでは，割り込みの発生と同時に，前もって登録しておいた割り込み処理ルーチンがただちに制御を受け，必要な処理を行なうことが許される。その処理が終り，割り込みが解かれた時点で，前の処理継続ができる。

非同期割り込み処理の登録は，ランゲージのレベルでは大きく２つのタイプに分けて考えることができる。その一つのタイプは，ＰＬ／ＩのＯＮステートメントに似たものに代表される。

```
ON  LITEPEN  BEGIN;
    Statement  1  ;
        ⋮
    Statement  n  ;
    END;
```

上のＯＮステートメントが実行された後に，ライトペン割り込みが発生すれば，実行中のステートメントはただちに中止され，Ｓｔａｔｅｍｅｎｔ １～ｎが実行されるであろう。

もう一つの割り込み処理登録のタイプは，明示的に割り込み制御表をこしらえておく方法である。このタイプでは，割り込みの種類と，これに対応する処理ルーチン名をあらかじめテーブルにして登

録しておき，割り込み発生に伴ってこのテーブルが参照され，対応する処理ルーチンが呼び出される。制御表作成は，次のようなサブルーチン・コールによって行なわれるであろう。

CALL SETUP （attntype(1), routine(1), ……
……attntype(n), routine(n)）；

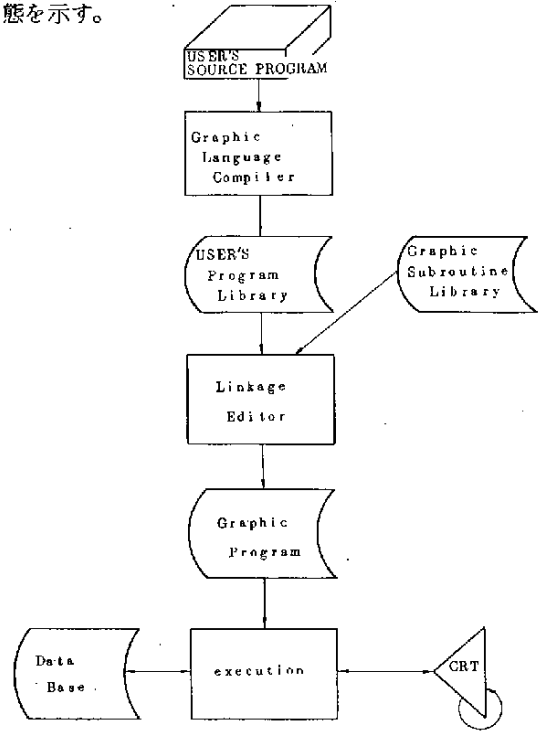
こうした，非同期処理を実現するランゲージがまた，PL／I や，ALGOLタイプで，ブロック構造を持っている場合には，割り込み登録もブロック制御を受ける仕様が取られる場合がある。すなわち，ある割り込み登録は，それが行なわれたブロック内のみに有効で，また，入れるになったブロック内では，内側ブロックにおける割り込み登録が，外側ブロックにおけるそれに優先する。— これは，同じくブロック構造を持つプログラミング・ランゲージにおける，変数の処理に良く似ている。— このような言語仕様の下では，処理の流れが，ブロックを通過する度の，割り込み制御表のスタック，デスタックが要求されるであろう。

また，実行中のステートメントの種類や，プログラムのステータスのいかんによっては，ある種の割り込みが禁止されたり，スタックされたりする機能を持たせるのも良い。

これら全てによって，完全な非同期割り込み処理が実現される。それは，マン・マシン・インタラクションの実現に画期的な手段を提供するであろう。

2.4.5 グラフィック・ランゲージの構造

これまでに記してきた諸機能を有する，グラフィック・ランゲージの全体像は，どのようになるであろうか。…… 次図に，その一形態を示す。



グラフィック・ランゲージ・ファシリティの特異性を最も象徴するのは恐らく，図中の"Graphic Subroutine Library"であろう。

複雑な図形処理を行なうプログラムでは，何度も述べたように，大量の，構造化されたデータ操作が必須である。さらに，グラフィック・CRTへの図形出力，割り込み解析の処理などの，やっかいなプログラミングの全てから，一般のグラフィック・プログラマは解放されるべきである。これらの処理は，コントロール・プログラムのマク

ロ・サポートとしてか，それとも，グラフィック・ランゲージ・ファシリティに付帯して提供されるサブルーチン・パッケージの任務として行なわれるべきである。ある機能を果すモジュールの要，不要を問わず，システムに固定される前者のアプローチより，必要に応じてモジュールがユーザのプログラムにつけ加えられる後者のアプローチの方が，一般的と考えられる。 ― 前者のアプローチはまた，既存のコントロール・プログラム・システムに大幅の改訂を加えなければならない困難がある，…… が，処理効率としては，このアプローチの方が，はるかに優れたものになる可能性を含んでいる。―

すなわち，後者のアプローチでは，Graphic Subroutine Library中に，

① データ・ストラクチュア・ハンドリング

② データ・ベース・マネージメント

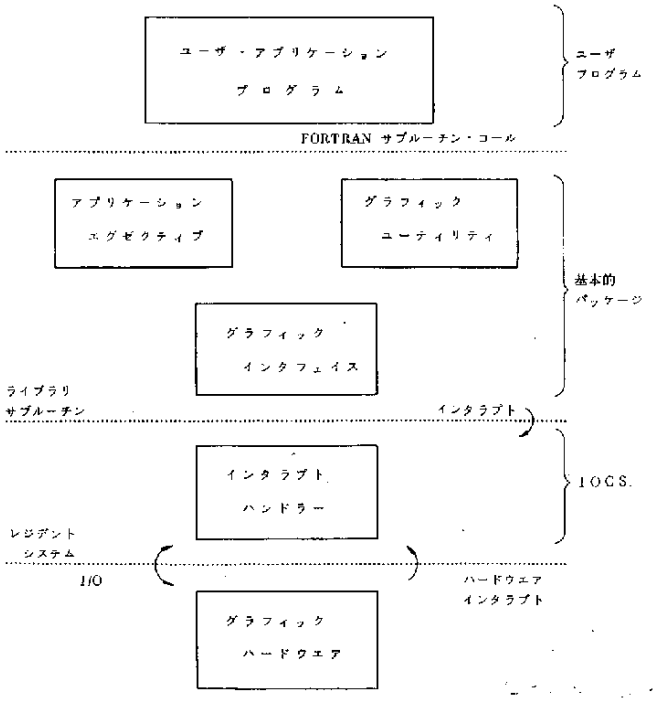
③ インタラクション・アナリシス

④ グラフィック・インターフェイス

等の機能を果たす，サブルーチン・パッケージが収容されるであろう。これらのサブルーチンに対する，ユーザ・プログラムのCalling sequence は，プログラマがそれを明示的に記述するのではなく，強力なステートメント群の翻訳結果として，グラフィック・ランゲージ・コンパイラが適宜に編集するべきものである。

次に実例として，CDC33／3500システムのMARK4．0というグラフィック・パッケージを取あげてみよう。このシステムのソフトウェア構成は次図のようになっている。

MARK4．0のソフトウエア・システム構成



この図からもわかるように，基本的なグラフィック・サポートのパッケージは次の３つに分かれている。

1. グラフィック・インタフェイス
2. グラフィック・ユーティリティ
3. アプリケーション・エグゼクティブ

まずグラフィック・インタフェイスはＭＡＳＴＥＲ（ＣＤＣ３３／３５００のモニタ）のモニタ・ステートで処理されるインタラプト・ハンドラとやりとりするいくつかのライブラリ・サブルーチン群から成っている。このグラフィック・インタフェイスは，グラフィックのハードウエアにかかわる一切の仕事をつかさどっている。すなわち，

イ．コンピュータの主メモリ上に組み立てられた図形情報等の，ディスプレイ・バッファ・メモリ上への転送およびその逆。

ロ．ディスプレイ・バッファ・メモリ上の内容のディスプレイ。

ハ．ディスプレイ・バッファ・メモリ上の内容の取扱い一切（読み取り，書き込み，バッファ・メモリ上での移動など）

ニ．ライトペン，コンソール・タイプライタ，ファンクション・キーなどから引き起こされる各種インタラプトの判別とその要求に対する処理。

ホ．図形，ライトペン，コンソール・タイプライタ，ファンクション・キー等への識別名の付与とその管理。

ヘ．トラッキング・クロスに関する一切の処理など。

ト．特殊なコントローラとあいまって，図形マクロ，図形サブルーチンの組立，保存，保守を行う。

このグラフィック・インタフェイスの内容は利用するコンピュータ・システム特にディスプレイのコントローラ，ディスプレイ装置等のハードウエアに深く依存する。そしてハードウエアの機能のすべてを，効果的に利用し得るようになっていなければならない。

次にグラフィック・ユーティリティは，大別して次の２つの機能を果たすライブラリ・サブルーチン群であり，各種のアプリケーション・プログラムから自由に利用できるようになっている。

イ．基本パターンのジェネレーションおよびディスプレイ

ロ．データ・マネージメント

上記イの範囲に入る機能が，普通狭い意味のコンピュータ・グラフィックスのソフトウエアと言われているように，かなり基本的なものである。これらのおもなものは，

- 点を表示するもの。
- 線分を描くもの。
- 円を描くもの。
- 拡大，縮小の機能。
- 文字や記号を描くもの。
- 図形マクロや図形サブルーチンを接続し表示するもの。

などである。この他図形の移動，回転，反転など，いろいろ考えられるが，どこまでシステムがサポートするかは，各システムの設計思想によるもので，数が多ければ多い程良いシステムであるとは言えない。基本的なものさえそろっていれば，あとはユーザの好みに応じて，組み合わせれば仕事が進められるからである。ＭＡＲＫ４．０ではこの機能として次の様なものが備えられている。

- 点の表示
- リセット（ビームの位置，性質のセット）
- 線分（実線，点線，鎖線，中心線）
- 円弧
- 拡大，縮小，移動，回転
- 図形マクロ，図形サブルーチン

これらの機能に加えてシザリングも非常に重要なものである。ＭＡＲＫ４．０では，普通のライン・シザリングの他に，アーク・シザリングという機能を有している。これは，与えられた矩形のフレームに，これも与えられた弧がぶつかるかどうかをチェックし，もし交点をもつとすれば何個所で交わり，かつそれらの座標値をプログラムに返してくれるものである。この機能を用いれば，倍率，移動量，回転角の変更等が便利に行える。

従来，このグラフィック・インタフェイスにあたるルーチンは，ほとんどソフトウエアで行ってきたものである。しかし，最近はマトリックス・マルチプライヤー（４元のマトリックス演算をハードウエアで高速に並行処理を行う装置），フリッピング・ディバイダー（ウィンドーイング，２次元表示座標への変換を行う装置），あるいは回転等の操作をハードウエアで行う等，各種のハードウエアが試作されているが，いづれもかなり高価であり，完全に汎用であるとも言い難い。

次に前記ロのデータ・マネージメントの機能についてであるが，これについては前節のデータ・ストラクチュアに詳しく述べられている。ＭＡＲＫ４．０においてはＰｌｅｘ ｄａｔａ ｓｔｒｕｃｔｕｒｅの概念を取り入れ，ユーザは任意の構造を持ったデータを，主コア・メモリ上，補助記憶のディスク上に関係なく，定義し，使用し，または更新したりすることができるようになっている。

ＣＤＣ．ＭＡＲＫ４．０のアプリケーション・エグゼクティブはかなり有力であり，ユーザはアプリケーション・プログラムをいくつかのタスクに分けて記述し，そのタスクのローディングとかその順序を制御するもので，これはＭＡＳＴＥＲに，特別なランダム・タスク・アクセスの機能をつけ加えたものである。

これにより，ユーザは種々のマニュアル操作，ＭＡＳＴＥＲのタスク呼出し，ライブラリ・サブルーチンの呼出し，などが自由に効率よく行うことができ，各種アプリケーションの作成に有力な武器となっている。

2.5 参 考 文 献

本章の全編を通じて

〔1〕 Principles of interactive systems
C. I. Johnson IBM Systems Journal Vol. 7, No. 3, 4 1969

から大きな示唆を受けた，……というより，この良くでき過ぎた論文から結局抜け出せなかった観すらある。この論文より出発して，様々の文献，資料に足を延ばして，コンピュータ・グラフィックスの世界をより詳しく知れば知る程，その広く透徹した洞察に頭が下がる思いがした。

第2節 インタラクティブ・ジョブ・プロセシングという面からは以下4編の論文を参考にした。
特に〔2〕はかなりの内容の濃い論文であろう。
その他ここには取りあげていないが，この報告書に参考文献としてあげてあるものの随所に参考となる意見が述べられている。

〔2〕 System considerations for Graphic data processing
E. M. Thomas (IBM) Computers and Automation 1967 11

〔3〕 A system for interactive graphical programming
William Newman 1968 SJCC

〔4〕 Conversational job control
S. H. Brown IBM Systems Journal Vol. 7, No. 3, 4 1969

〔5〕 Conversational display capability
F. W. Gagliano 他 上記と同じ

第3節 データ・ストラクチュアと，第4節 ランゲージ・ファシリティに関しては，共通に役に立った論文が多かったが，特にデータ・ストラクチュアの面からは，

〔6〕 Compound data structure for computer aided design
J. C. Gray Proceedings A. C. M. National Meeting 1967

に，各種のデータ・ストラクチュアの鳥瞰図的記述がされており，最初のアプローチに有効であった。

データ・ストラクチュアの概念を摑むには他に，

〔7〕 Computer graphics software techniques
S. Bird Computer Graphics Plenum Press 1969

〔8〕 Theoretical foundations for the computer aided design system
D. T. Ross and J. E. Rodriguez AFIPS SJCC 1963

〔 9〕 Data structure in two level store
C. A. R. Hoare — Proceedings A. C. M. National Meeting 1968

〔10〕 SKETCHPAD - A man-machine Graphical communication system
I. E. Sutherland — AFIPS SJCC 1963

〔11〕 Graphical communicational and control language
L. G. Roberts — The Computer Display Review Adams Associates Inc.

等によった。

また，各種データ・ストラクチュア及びランゲージの実際を理解する上で，CORALに関しては，

〔12〕 The CORAL Language and data structure
W. R. Sutherland — 上掲書

APLについては，

〔13〕 APL-A language for associative data handling in PL/I
G. G. Dodd — AFIPS FJCC 1966

AEDは，

〔14〕 The AED approach to generalized computer-aided design
D. T. Ross — Proceedings A. C. M. National Meeting 1967

〔15〕 The AED free storage package
D. T. Ross — Communications of the ACM Vol. 10, No. 8 August 1967

LEAPは，

〔16〕 The LEAP language and data structure
P. D. Rovner and J. A. Feldman — Proceedings of IFIP congress 68

〔17〕 An ALGOL-based associative language
J. A. Feldman — Communications of the A. C. M Vol. 12, No. 8 August 1968

ASPは，〔 6 〕と，

〔18〕 ASP-A ring inflemented associative programming package
J. C. Gray — Communications of the A. C. M. Vol. 11, No. 3, August 1968

また，DSPLに関しては，

〔19〕 Data structure programming language
D. S. Evans and Van Dam Proceedings of IFIP Congress 68

等を参照した。

データ・ストラクチュアを論ずる上で，その他に参照した文献としては，

〔20〕 Auxiliary - storage associative data structure for PL/I
A. J. Symonds I. B. M. System Journal （前出）

〔21〕 Scatter Storage Technique
R. Morris Communications of the A. C. M.
Vol. 11, No. 1, January 1968

〔22〕 An improved hash code for scatter storage
W. D. Manrer 同上

〔23〕 A multilebel modeling structure for interactive graphic design
H. B. Baskin and S. P. Morse
I. B. M. Systems Journal （前出）

24〕 Data structure and techniques for remote computer graphics
I. W. Cotton and F. S. Greatorex J R.
AFIPS FJCC 1969

〔25〕 Data structure and man-machine communications for network problems
D. S. Evance and J. Katzenelson
I. E. E. E. Vol. 55, No. 7, 1967

〔26〕 A compact data structure for computer-aided design
A. D. Dam and D. S. Evance
AFLPS SJCC 1967

〔27〕 Graphic data management in a time-shared environment
S. Bowman and R. A. Lickhalten
AFIPS SJCC 1968

〔28〕 A ring structure processor for a small computer
N. E. Wiseman and J. O. Hiles
The Computer Journal, February 1968

〔29〕 A note on compiling display file from a data structure
N. E. Wiseman The Computer Journal, August 1968

〔30〕 The construction of hierachic and non-hierarchic classification
N. Jardine and R. Sibson The Computer Journal 1968

等があり，〔２０〕，〔２１〕，〔２２〕は hashing technique に近づく上で，特に役立った。

グラフィック・ランゲージへの考察で示唆をあおいだ文献として，〔１２〕～〔１９〕の他に，

〔31〕 A system for implementing interactive applications
F.C. Chen and R. L. Dougherty
I.B.M. Systems Journal （前出）

〔32〕 A subroutine package for FORTRAN
A.D. Rully 同上

〔33〕 A language for three dimmensional geometry (GL)
P.G. Comba 同上

〔34〕 A general purpose graphic language
H.E. Kulsrud Communications of the A.C.M.
Vol. 11, No. 4, 1968

〔35〕 A problem oriented graphic language
P.M. Schwin Proceedings of the A.C.M. National Conference 1967

〔36〕 A non-procedural language for retrieving information from data bases
T.W. Olle Proceedings of the IFIP Congress '68

〔37〕 The control of exceptional conditions in PL/I object programs
J. M. Noble 同上

〔38〕 A two-dimensional programming system
M.I. Bernstein
T.G. Williams 同上

〔39〕 A system for interactive graphical programming
W.M. Newman AFIPS SJCC 1968

〔40〕 A language and model for computer-aided design
N.J. Denil Proceedings A.C.M. National Meeting 1966

〔41〕 Graphic language translation with a language independent processor
R.A. Horrison AFIPS FJCC 1967

〔42〕 SPARTA, a procedure oriented programming language for the manipulation of arbitrary line drawings
L. Mezei Proceedings of the IFIP Congress 68

等をあげておく。

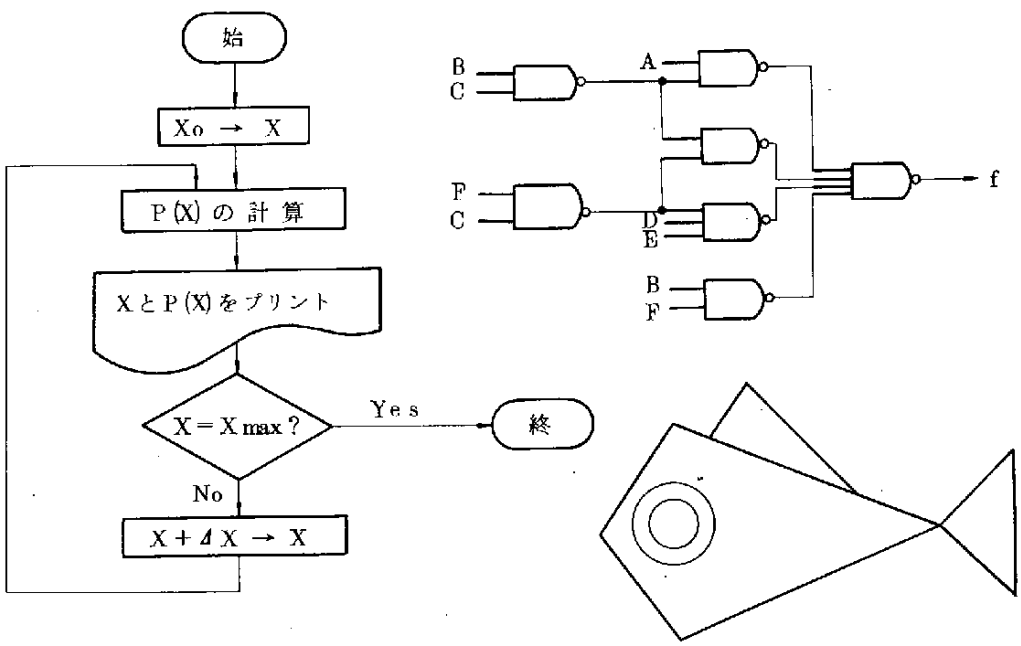
3．コンピュータ　グラフィックスにおける数学的手法

Computer Graphicsというのはコンピュータを用いて図形処理を行うものであるが，そのシステムやアルゴリズムを設計する際に図形的センス――図形の性質やその処理法に対する慣れ――を身につけることが望まれる。というのは図形の性質を上手に利用することによって良いプログラムを作成することができるからである。そうは言っても，あくまでもコンピュータによる図形処理であるから，これにうまくのらない（例えば理論的には美しくてもコンピュータにやらせると時間がかかることなど）様なアルゴリズムではいただけない。この辺がComputer Graphicsのユニークな所であり，コンピュータを用いる数値計算法と共通な特質を有している。すなわちコンピュータの資質を生かし，しかも図形の性質を上手に利用して処理を行うという両方にマッチしたものが要求されるわけである。

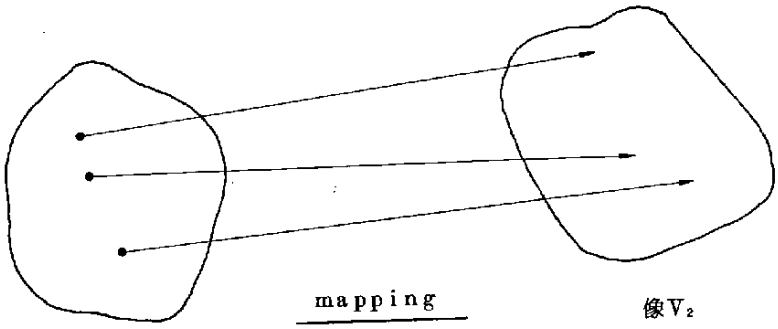
人間と図形との交りは古く，現在においても図形は人間生活にますます密接し，重要なものとなって来ている。現在までに，図形の研究は多くの先生によって受け継がれいろいろな幾何学が生まれて来た。その手法もさまざまであり，その中にはComputer Graphicsにおける図形処理手法として欠くことのできない解析的な方法，すなわち座標の導入，パラメトリック表示法，マトリックス，ベクトル解析，テンソル解析などが含まれている。

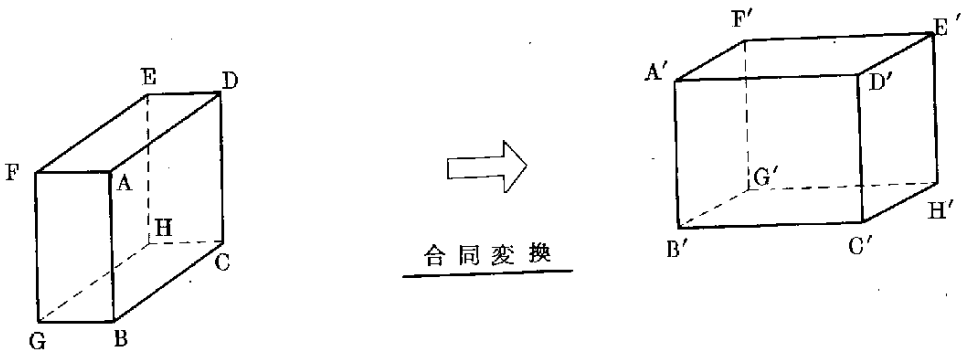
本章は，その中で特に基礎的な数学的手法の紹介を行う。こゝでとりあげるのはPICTURE CALCULUS，投象問題，Fitting・Smoothing問題，およびHidden line elimination 問題であり，これらの問題の意味や手法を次に簡単に述べる。

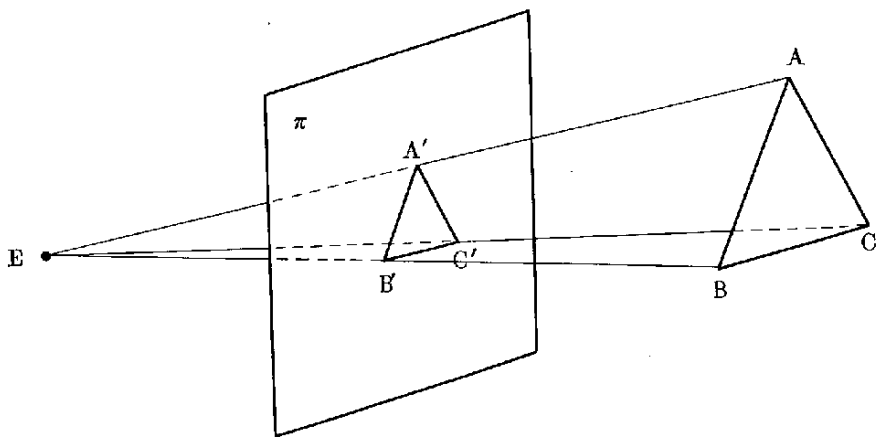
PICTURE CALCULUS；線図，Flow Chartの様にシンボル図形とそれらの結びつきが問題となる様な図を扱う手法の一つである。その発想は一つ一つのシンボルを文字におきかえ，結びつきを文字式に表わしてコンピュータに処理をさせようというもので，図形言語と深いつながりを有する。



投象問題；従来，図学などによって行われていた各種投象図を座標変換によって解析的にコンピュータに行わしめようとするものであるが，本質的には写像問題に帰着される。３次元の空間図形Ｖを平面図形Ｖ′に写像（mapping）したものが投象図であるが，３次元の空間図形V_1を合同変換（回転，平行移動，裏返し，それらの合成）を施こして得られるV_2もV_1の写像と考えることができる。



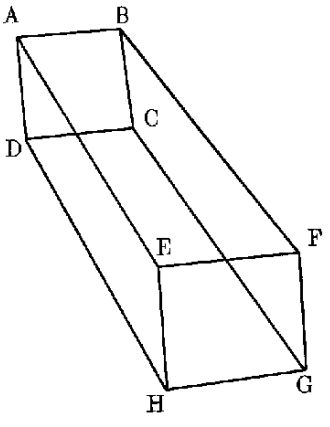




曲線，曲面に於けるＦｉｔｔｉｎｇ およびSmoothing問題；Ｆｉｔｔｉｎｇ とは与えられた点集合（離散点）を通り，なめらかな曲線または曲面を作ること，Ｓｍｏｏｔｈｉｎｇとは与えられた点集合の各点の許容された近傍を通りなめらかな曲線または曲面を作ることである。この問題は一種の補間法（ｉｎｔｅｒｐａｌａｔｉｏｎ）の問題であるが，特性近似問題とも関連がある。現在，航空機産業や自動車工業におけるＣＡＤの一部すなわち自由曲面などの設計などで大いに開発が進んでいるが，理論的には体系づけられていない。将来は，自由曲面の創成だけでなく各分野で用いられるいろいろな補間法が提案され開発されていくとともに，補間法全体としてもまとまってくるであろう。

隠れ線消去ｈｉｄｄｅｎ ｌｉｎｅ Ｅｌｉｍｉｎａｔｉｏｎ の問題；たとえば直方体の頂点の座標を計算機に入力し適当なスケーリングを行いプロッターにその透視図を画かせたとする。隠れ線消去の機能がないと，プロッターは面によって隠される陵まで画き出してしまい得られた図形はいわゆるｗｉｒｅ ｓｔｒａｃｔｕｒｅになる。本来ならば，陵ＢＣ，ＤＣ，ＣＧは面によって隠されるので破線あるいは全然画かないことが望ましい。複雑な図形の場合特にこの機能が必要になって来る。

右図の様な凸多面体の場合は簡単なベクトル計算によってｈｉｄｄｅｎ ｌｉｎｅ を消すことが出来るのであるが，凹凸のある多面体や曲面体の場合は一般に簡単ではない。この種の問題は結局図形と図形との相互関係をどの様に評価するかということである。たとえば２本のベクトルが平行であること，１つの図形に他の図形がふくまれることなどをどう評価してコンピュータに判断させるかということである。更に進んで透視図に陰影をつけたり，着色したりすることもこの問題の範囲にはいるのである。



Wire Structure

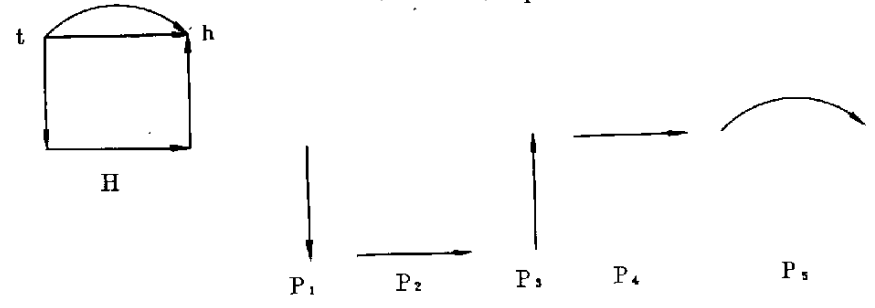
3.1 PICTURE CALCULUS

我々の取り扱う図形は大きく分けて2つある。いわゆるnatural picturesとartifical picturesである。前者は空間図形やその投象図などに代表され，後者の代表はgraph（トポロジー的な）である。line drawings flow charts, block letters などもこの類に属する。natural pictures の処理にはパラメトリック方程式や座標変換などが活躍するがartifical pictures ではここで紹介するpicture calculusを中核とした言語でこれらを処理する。しかし実際の図形処理においては，これらの2種類の処理法を適当に使いわけて行うことになると思う。ここで紹介するものはW. F. Miller and Alan C. Shaw "A Picture Calculus", Stnford University, Stanford, Califarnia によった。picture calculusと言語システムとの関連は文献その他を参照されたい。

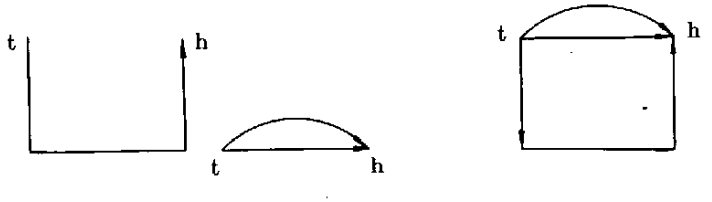
picture calculusは扱う問題によってルールが異るものである。したがってここで紹介するcalculus はその一例となる。

(i) Primitives；picture あるいはgraphを構成する適当な単位を"primitive" と称する。ここで「適当な」というのは取り扱う場所，時によってある場合は一本のベクトルであったり，またある場合は或る簡単なpicture であったりするからである。しかしながら，少くともTailとHead が決まっている必要がある。なぜなら，TailとHead を通じてたくさんのPrimitiveがつながっているからである。

（例）　左図のHの線図は次の様なprimitiveからなっていると考えられる



(ii) ルール；Primitiveがいくつか（一つでもよい）集まってgraphを構成する。graphは処理言語ではsentence といっている。上の例ではprimitive自身もgraphであるが，その他下図に示すものもgraph である。

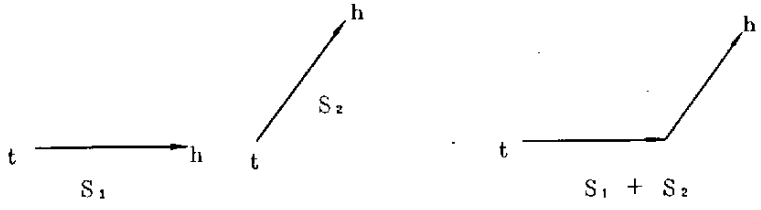


graphにおいてもTAIL, HEADが存在することに注意すること。

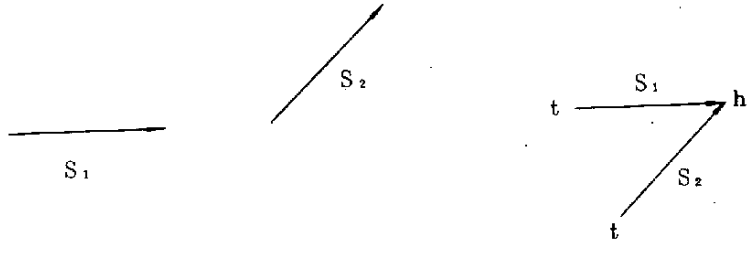
(イ) binary operator(+, −, ×, ∗, ～)に関するルール

以後graphを S_i とする。

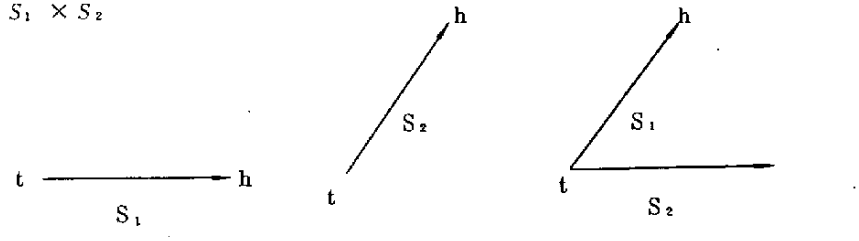
・$S_1 + S_2$



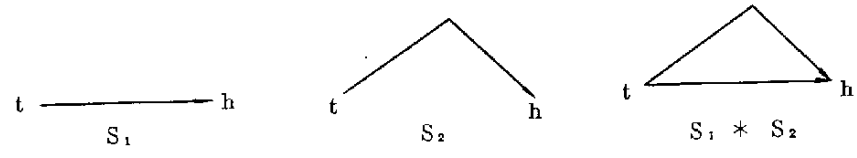
・$S_1 - S_2$



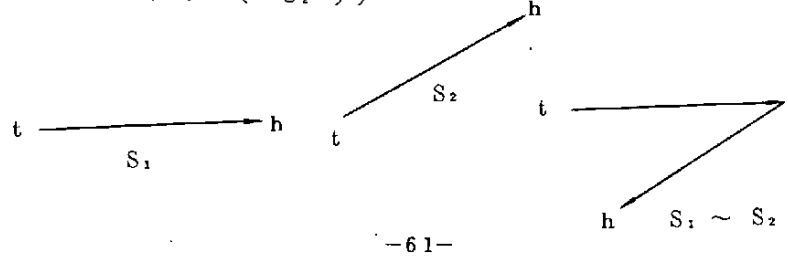
・$S_1 \times S_2$



・$S_1 * S_2$



・$S_1 \sim S_2 \equiv (S_1 + (\sim S_2))$



(ロ) TAILとHEADの規則

TAIL(S_1 θ S_2)=TAIL(S_1)

HEAD(S_1 θ S_2)=HEAD(S_2)

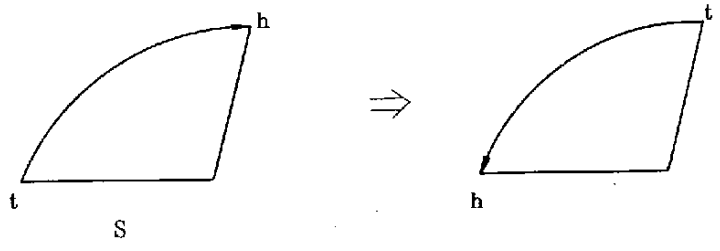
θ ; +, ×, −, *, ～

(ハ) unary operater (～, ⌐)およびT(w)(アフィン変換)

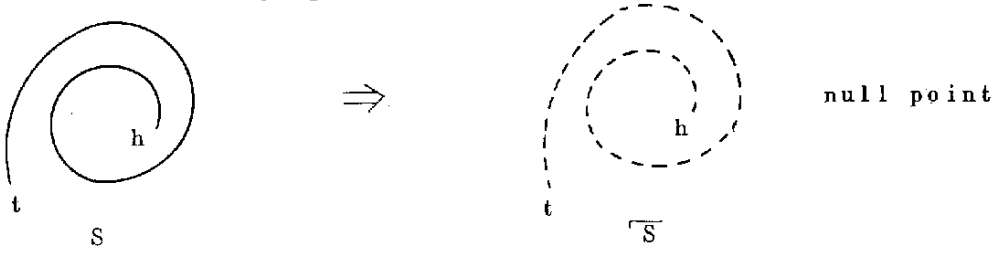
・～operater; HEADとTAILを取りかえる変換

TAIL(～S)=HEAD(S)

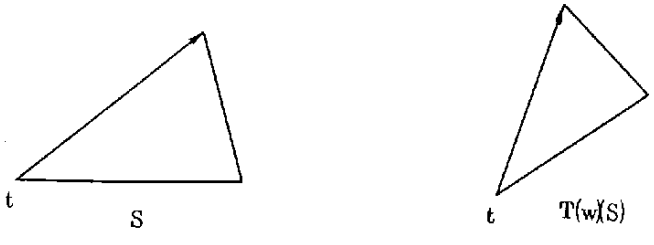
HEAD(～S)=TAIL(S)



・⌐operater; graph の消去

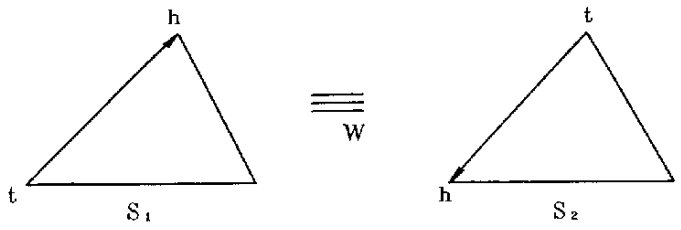


T(w)



(iii) graphの等価; 最初に述べたようにcalculus にはいろいろ考えられるが, その相異を生ずる大きな原因の1つはgraphの等価をどの様に定義しているかということである。ここでは, PDL言語(Picture DescriptionLanguage)で採用されているものを紹介する。

(イ) Weak Equivalence, Equivalence



形としてはS_1 とS_2 は合同であるがＴＡＩＬ，ＨＥＡＤの位置がちがう場合，ｇｒａｐｈS_1 とｇｒａｐｈS_2 とは，Ｗｅａｋ Ｅｑｕｉｖａｌｅｎｃｅと言って，次の様に書く。

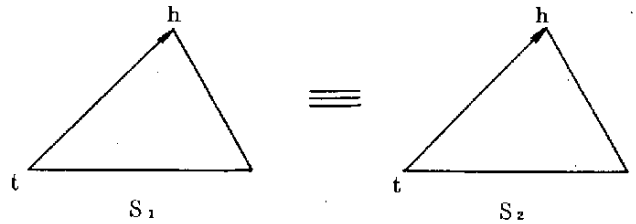
$S_1 \equiv_W S_2$

一方，S_1 とS_2 がEquivalence $S_1 \equiv S_2$ というのは次の条件を満すときを言う。

$S_1 \equiv_W S_2$

ＴＡＩＬ(S_1)＝ＴＡＩＬ(S_2)

ＨＥＡＤ(S_1)＝ＨＥＡＤ(S_2)



(Ⅳ) 等価が定義されたので演算則を述べる。

(イ) 結合法則

(1) $(S_1 + S_2) + S_3 \equiv S_1 + (S_2 + S_3)$

(2) $(S_1 \times S_2) \times S_3 \equiv S_1 \times (S_2 \times S_3)$

(3) $(S_1 - S_2) - S_3 \equiv S_1 - (S_2 - S_3)$ ⇐ 代数とはちがうことに注意

(4) $(S_1 * S_2) * S_3 \equiv S_1 * (S_2 * S_3)$

(ロ) 交換法則

(1) $S_1 * S_2 \equiv S_2 * S_1$

(2) $S_1 \times S_2 \equiv_W S_2 \times S_1$

$S_1 - S_2 \equiv_W S_2 - S_1$

(ハ) ～Ｏｐｅｒａｔｏｒ

(1) $\sim(S_1 + S_2) \equiv (\sim S_2) + (\sim S_1)$

$\sim(S_1 * S_2) \equiv (\sim S_2) * (\sim S_1)$

(2) $\sim(S_1 \times S_2) \equiv (\sim S_2) - (\sim S_1)$

$\sim(S_1 - S_2) \equiv (\sim S_2) \times (\sim S_1)$

(ニ) Ｂｌａｎｋｉｎｇ Ｏｐｅｒａｔｏｒ ‾

(1) $\overline{(\overline{S})} \equiv \overline{S}$

(2) $\overline{(S_1 \ \theta \ S_2)} \equiv (\overline{S_1}) \ \theta \ (\overline{S_2})$ 　θ ; +, ×, −, ＊

(3) $\overline{(\sim S)} \equiv \sim (\overline{S})$

(ホ) Null Point Primitive

S, λ をNull Point Primitiveとすれば以下が成立する。

(1) $S \ \theta \ \lambda \equiv S \ \theta \ \lambda$

(2) $S \ \varphi \ \lambda \equiv S$

(3) $\sim \lambda \equiv \lambda$

(4) $\lambda \ \theta \ \lambda \equiv \lambda$

但し θ ; +, ×, −, ＊

φ ; +, ×, −

3.2 投 象 問 題

空間に存在する図形または立体を平面に写像（mapping）して得られた像を投象図または投象（projection）と言っている。立体図学によれば投象は次の様に分けられる。

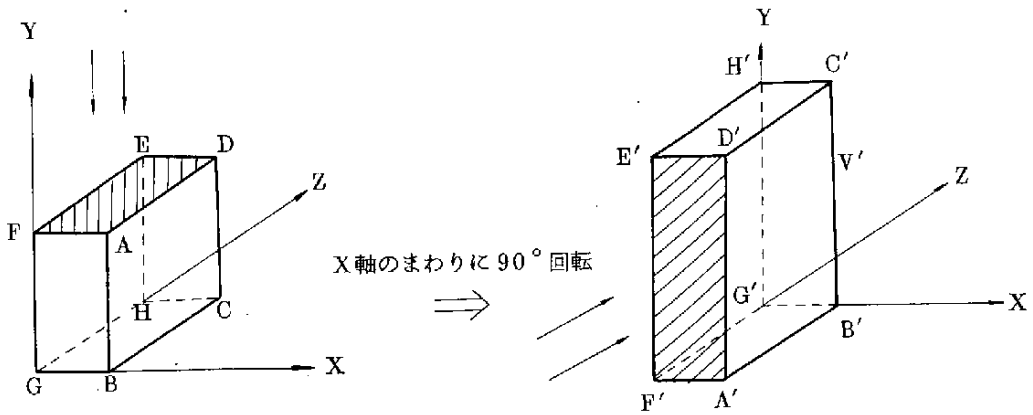
- 投象
  - 透視投象
  - 平行投象
    - 直投象
      - 正投象
      - 軸測投象
        - 軸測投象
        - 等測投象
      - 標高投象
    - 斜投象
      - キャバリア投象（Cavalier projection）
      - キャビネット投象（Cabinet projection）

Computer Graphicsでは，パラメトリック方程式や座標の形で格納されている空間図形を解析的（座標変換）に透視投象，正投象を得ようとするものである。解析的というのは，第1節で述べたようにmapping という発想法から出ている。ここでは正投象・透視投象において用いられているComputer Graphicsにおける投象処理を紹介する。なお本節を書くにあたっては，D. V. Ahuja and S. A. Coons, "Geometry for construction and display," IBM Systems Journal vol 7／Number 3&4 に負うところが大である。

Computer graphicsに於ける投象の処理は，座標変換（4×4 matrix の計算）によって行われる。すなわち所望の投象を得るには，同次座標で記述された空間図形に座標変換（所望の投象によって決まる）を施こしpicture plane上の座標に変換し，その座標をgenerationするのである。以下具体例を通して説明していきたい。

＜例1＞ 直方体の正投象 ── 左下図の様な直方体ABCDEFGが存在していて，XY平面をpicture planeとしたとき，立面図，平面図，側面図を求める問題を考える。

A（1, 2, 0），B（1, 0, 0），C（1, 0, 3），D（1, 2, 3），
E（0, 2, 3），F（0, 2, 0），G（0, 0, 0）



(i) TOP View （立面図）

上から見た投象図を得るには次の様にする。XY平面がpicture planeであるから，直方体VをX軸のまわりに90°回転する。

$$\underset{D}{[\,1 \quad 2 \quad 3 \quad 1\,]} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \underset{D'}{[\,1 \quad 3 \quad -2 \quad 1\,]}$$

各点も同様に変化をうける。直方体V′をZ軸に沿った平行線を矢印のようにあててXY平面上に投象図を作ればよい。

XY平面上の座標に変換するには次のマトリックスを座標に乗ずればよい。

$$\text{Projection Matrix;} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\underset{D'}{[\,1 \quad 3 \quad -2 \quad 1\,]} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \underset{D''}{[\,1 \quad 3 \quad 0 \quad 1\,]}$$

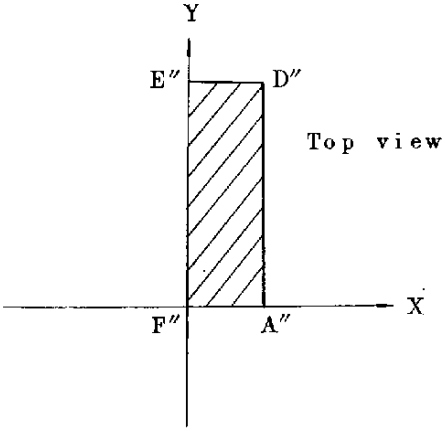
D″（1, 3, 0）がgenerationのための座標ということになる。

各点も同様にしてgenerationのための座標を得ることができる。

〔1, 2, 0, 1〕R×P＝〔1, 0, 0, 1〕；A→A″

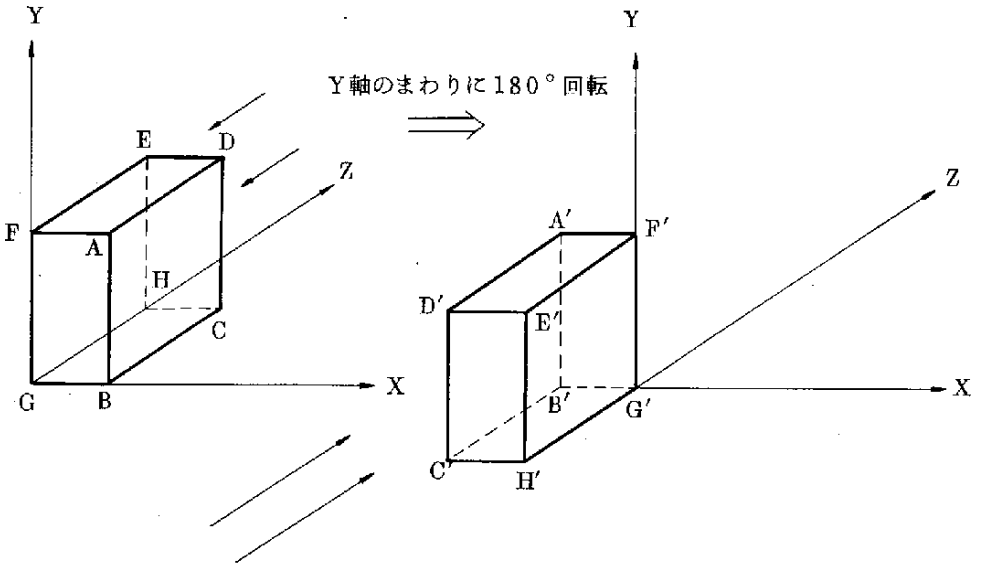
〔0, 2, 3, 1〕R×P＝〔0, 3, 0, 1〕；E→E″

〔0, 2, 0, 1〕R×P＝〔0, 0, 0, 1〕；F→F′



(ii) Front view

TOP view, Righ side view と同様にPicture planeがＸＹ平面，平行光線がＺ軸に沿って正の向きに当っているとしたときの(ii)のFront viewを求める方法を考える。



回転マトリックスは$\theta = 180°$を代入して

$$R = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

を得る。

故に $[1, 0, 3, 1]R = [-1 \quad 0 \quad -3 \quad 1]$ C→C′

$[1, 2, 3, 1]R = [-1 \quad 2 \quad -3 \quad 1]$ D→D′

$[0, 2, 3, 1]R = [0 \quad 2 \quad -3 \quad 1]$ E→E′

$[0, 0, 3, 1]R = [0 \quad 0 \quad -3 \quad 1]$ H→H′

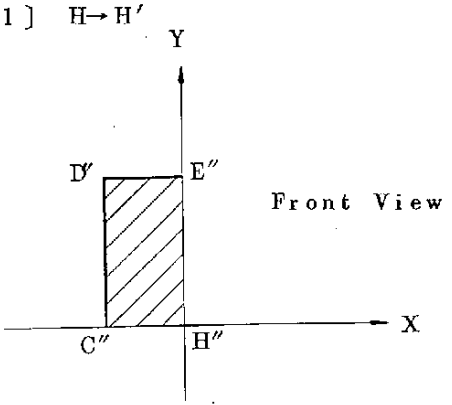
これらの座標に投象マトリックスを乗ずると

$[-1 \quad 0 \quad 0 \quad 1] = C''$

$[-1 \quad 2 \quad 0 \quad 1] = D''$

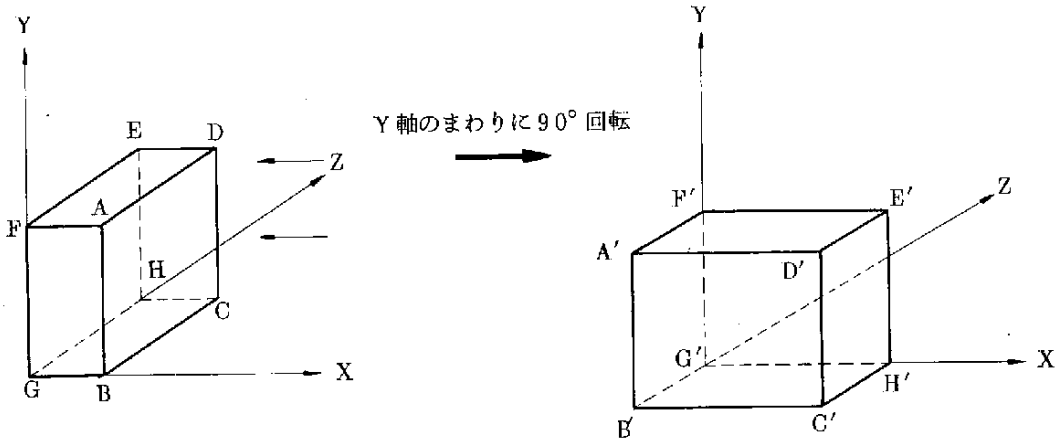
$[0 \quad 2 \quad 0 \quad 1] = E''$

$[0 \quad 0 \quad 0 \quad 1] = H''$



(iii) Right−side view

X軸に沿って負方向に平行線をあててＹＺ平面に投象した図を得たい場合

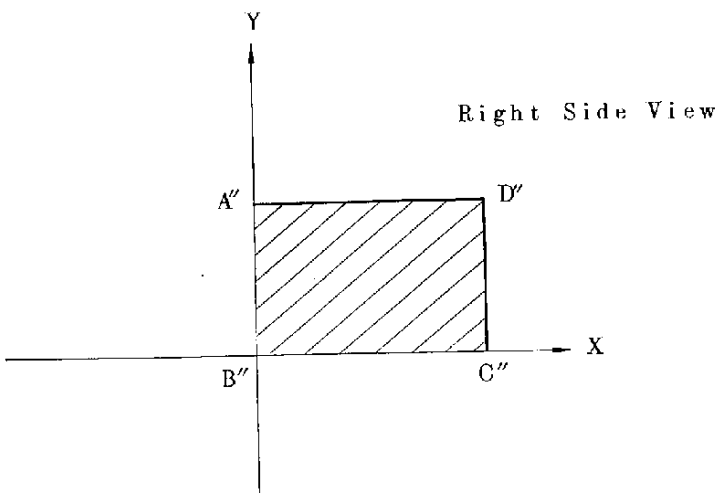


Right side view はＶをＹ軸のまわりに９０° 回転したＶ′ にＺ軸に沿って正方向に平行光線を当てＸＹ平面に投象したものである。

(i)と同様に

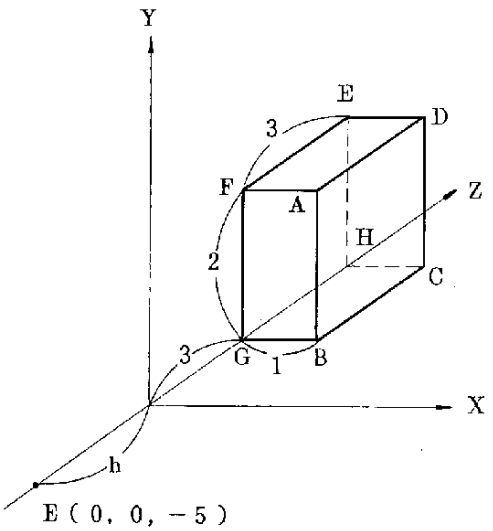
$$\underset{D}{\begin{bmatrix} 1 & 2 & 3 & 1 \end{bmatrix}} \begin{bmatrix} 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times P = \underset{D''}{\begin{bmatrix} 3 & 2 & 0 & 0 \end{bmatrix}}$$

D以外の頂点においても同じように変換して下の投象図を得る。



≪ 例２ ≫ 直方体の透視投象 ── 左下図の様な直方体Ｖの透視投象（Ｚ軸に沿う）を求む。ただし視点Ｅの位置は（０，　０，　－５）とする（ｈ＝５）

A (1, 2, 3), B (1, 0, 3), C (1, 0, 6)
D (1, 2, 6), E (0, 2, 6), F (0, 2, 3)
G (0, 0, 3), H (0, 0, 6)



E (0, 0, −5)

$$[\,1,\ 2,\ 3,\ 1\,]\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1/5\\0&0&0&1\end{pmatrix}=[\,1,\ 2,\ 0,\ 8/5\,]\sim[\,0.63,\ 1.3,\ 0,\ 1\,]$$

S

A → A″

$[\,1,\ 0,\ 3,\ 1\,]S=[\,1,\ 0,\ 0,\ 8/5\,]\sim[\,0.63,\ 0,\ 0,\ 1\,]$

B → B″

$[\,0,\ 2,\ 3,\ 1\,]S=[\,0,\ 2,\ 0,\ 8/5\,]\sim[\,0,\ 1.3,\ 0,\ 1\,]$

F → F″

$[\,0,\ 0,\ 3,\ 1\,]S=[\,0,\ 0,\ 0,\ 8/5\,]\sim[\,0,\ 0,\ 0,\ 1\,]$

G → G″

$[\,1,\ 0,\ 6,\ 1\,]S=[\,1,\ 0,\ 0,11/5\,]\sim[\,0.46,\ 0,\ 0,\ 1\,]$

C → C″

$[\,1,\ 2,\ 6,\ 1\,]S=[\,1,\ 2,\ 0,11/5\,]\sim[\,0.46,\ 0.9,\ 0,\ 1\,]$

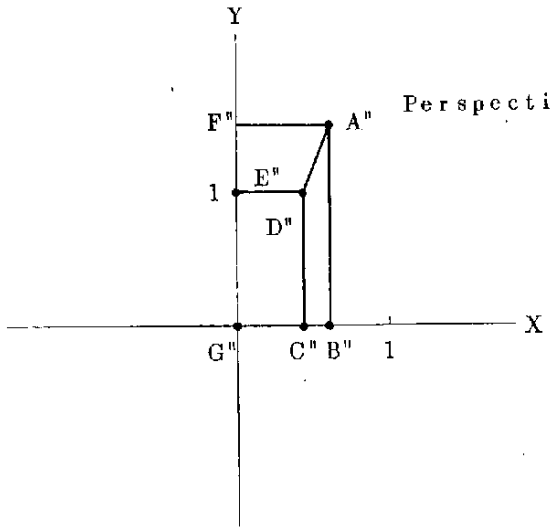
D → D″

$[\,0,\ 2,\ 6,\ 1\,]S=[\,0,\ 2,\ 0,11/5\,]\sim[\,0,\ 0.9,\ 0,\ 1\,]$

E → E″

$[\,0,\ 0,\ 6,\ 1\,]S=[\,0,\ 0,\ 0,11/5\,]\sim[\,0,\ 0,\ 0,\ 1\,]$

H → H″



以上直方体の正投象および透視図の処理法をのべたが，ここであらわれた同次座標，Projection Matrix, Perspective Matrix, 回転，などのいろいろな同次座標を変換するマトリックスについて簡単にのべよう。

- 同次座標（homogeneous coordinates）

  空間の一点Pの同次座標とは次の様なものを言う。

  P点の直交座標〔 x, y, z 〕（これを同次座標に対して"ordinary coordinates"と言っている）に対応して〔 wx, wy, wz, w 〕をP点の同次座標（homogeneous coordinates）と言う。

  通常座標との関係は $x=\frac{(wx)}{w}$, $y=\frac{(wy)}{w}$, $z=\frac{(wz)}{w}$ で換算できる。すなわち第4要素で第1，第2，第3要素を割れば普通の座標がえられる。P点の同次座標をP，Q点の同次座標をQとすれば

  $P=\alpha Q$　　α；スカラー

  であるときは実はP点とQ点は同一点であることに注意する。例えば〔 2, 3, 4, 1 〕と〔 −4, −6, −8, −2 〕は同じ点をあらわす。

- 同次座標の変換マトリックス

1. 平行移動（Translation）

$$\begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ e & f & g & 1 \end{bmatrix} = \begin{bmatrix} x+e & y+f & z+g & 1 \end{bmatrix}$$

2. スケーリング（一様な拡大または縮小を行う）

$$[\,x \quad y \quad z \quad 1\,]\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & S\end{bmatrix} = [\,x \quad y \quad z \quad S\,] \sim [\,x/S \quad y/S \quad z/S \quad 1\,]$$

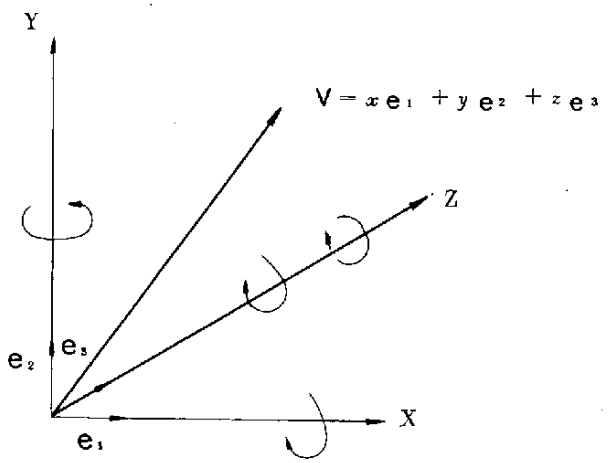
$S > 1$　　縮小

$S = 1$　　もとのまま

$S < 1$　　拡大

$1/S$ が拡大率になることに注意されたい。

3. 回転（Rotation）



V^x, V^y, V^z はそれぞれベクトルVをX軸，Y軸，Z軸のまわりに正の向きに θ だけ回転したベクトルとする。

(i) V^x

x 成分；変化なし

y 成分；$y\,e_2$ はY軸上のベクトルである。これがX軸のまわりに θ だけ回転するのであるから

$$y\,e_2 \Rightarrow y\cos\theta\,e_2 - y\sin\theta\,e_3$$

同様に　$z\,e_3 \Rightarrow z\sin\theta\,e_2 + z\cos\theta\,e_3$

故に　$y\cos\theta + z\sin\theta$

z 成分；$-y\sin\theta + z\cos\theta$

故にX軸のまわりの回転は次の様になる。

X軸

$$V \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = V^{x}$$

(ii) V^{y}

y 成分；変化なし

z 成分；$x\sin\theta + z\cos\theta$

x 成分；$x\cos\theta - z\sin\theta$

Y軸

$$V \times \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = V^{y}$$

(iii) V^{z}

z 成分；変化なし

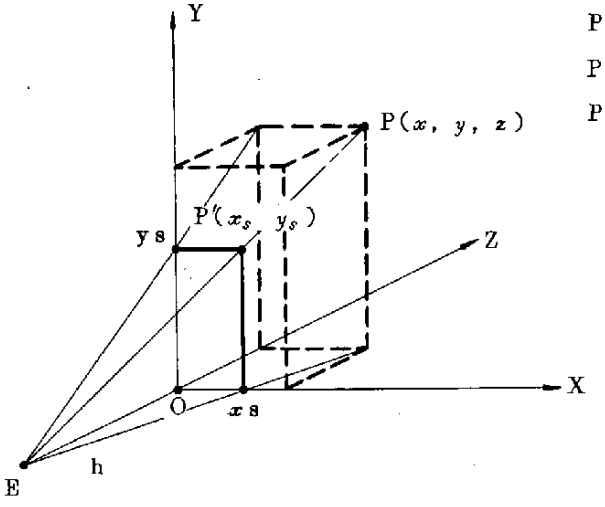
x 成分；$x\cos\theta + y\sin\theta$

y 成分；$-x\sin\theta + y\cos\theta$

Z軸

$$V \times \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = V^{z}$$

4. 透視投象変換



$P' = [x_s, y_s, 0, 1]$

P；原像

P'；投像

$x_s : x = h : (h+z) = y_s : y$

$$\therefore \quad x_s = \frac{hx}{h+z} = \frac{x}{1+\frac{z}{h}}, \quad y_s = \frac{y}{1+\frac{z}{h}}, \quad z_s = 0$$

$$\begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1/h \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} x & y & 0 & 1+\frac{z}{h} \end{bmatrix}$$

$$\sim \begin{bmatrix} \frac{x}{1+\frac{z}{h}} & \frac{y}{1+\frac{z}{h}} & 0 & 1 \end{bmatrix}$$

h を 1 に比して十分大きくなると正投象変換マトリックスになる。

5. 正投象変換

$$\begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} x & y & 0 & 1 \end{bmatrix}$$

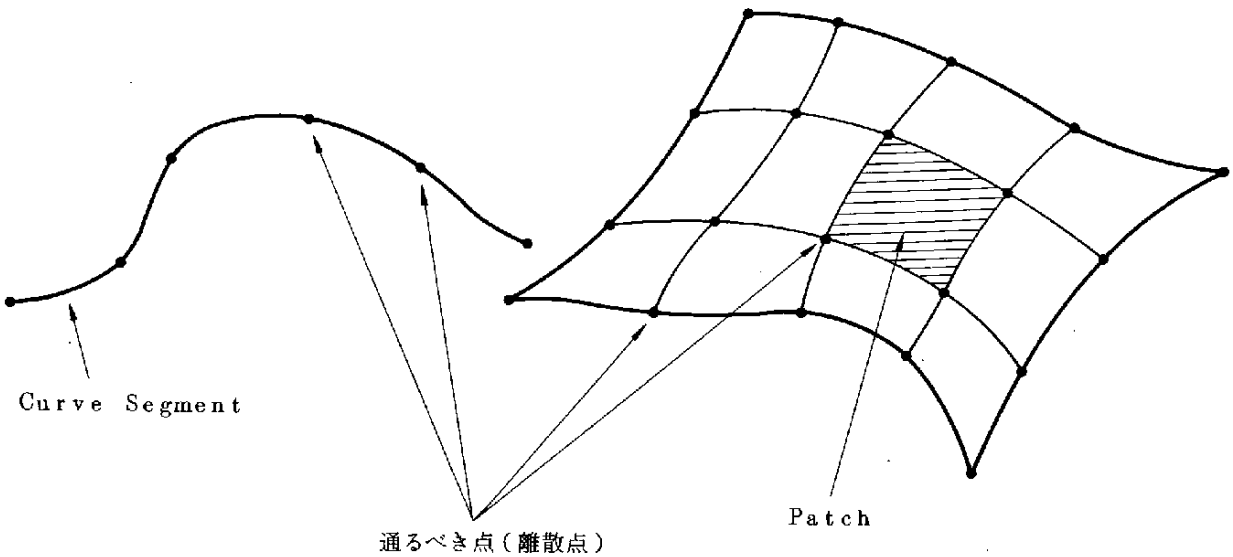
3.3 曲線・曲面のFittingとSmoothing

Computer Graphicsにおける重要な技術の一つにFitting およびSmoothingがある。これは第1節で述べた様に

(i) Fitting（適合，またはあてはめ）；与えられた点集合（離散点列）を通り，なめらかな曲線または曲面を作ること

(ii) Smoothing（平滑化）；与えられた点集合の各点に許容された近傍を通りなめらかな曲線または曲面を作ること

であると言われている。(i)でも(ii)でも手法としては曲線要素Curve Segment，曲面要素Surface Segment（Patch とも言う）を与えられた離散点から補間を行って順次決定しこれらをつないで全体の曲線や曲面を作り上げてゆくものである。ただ異る点は，(i)は曲線または曲面が各点を確実に通ることが必要であり，(ii)では各点の予め許容された近傍を通るということである。



一般に自由曲線や自由曲面の取り扱いには一変数パラメータt，二変数パラメータu，vを用いたベクトル方程式が工学的に有効とされている。この立場に立ってCurve Segment，Surface Segment の定義式，Segment 間の接続法，曲線における変曲点・曲面における凹凸の改善，などがいろいろ提案され実際に行われているが，いまだ定着していない様である。この種の問題でそのアルゴリズムの評価の規準となるのは，(a)表現能力の範囲 (b)出来あがった図形の連続性の程度 (c)局所的な修正可能か否か (d)計算時間 などである。

Fitting, Smoothingの手法は数多くあるが，本節では，曲線，曲面のFitting の一方法を紹介する。曲線については，D. V. Ahuja, "An algorithm for generating spline-like curves" IBM Systems Journal, volume Seven /Numbers Three & Four /1968 を，曲面については日本電子工業振興協会，「機械設計のためのプログラムの研究」昭和43.2月5章を紹介する。Fitting, Smoothingの手法はこの他にも多くありそれらについては参考文献を参照されたい。

(I) Curve Fitting(Spline-like curve)

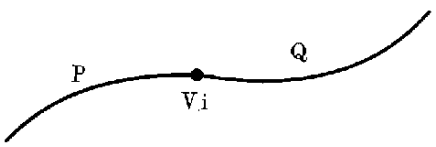
本方法は，3次元空間におけるn個の点列を以下の条件に従ってCurve Fittingしていくものである。

① Curve Segmentの定義式（補間式）

$$x=\frac{a_1u^3+b_1u^2+c_1u+d_1}{au^3+bu^2+cu+d}$$
$$y=\frac{a_2u^3+b_2u^2+c_2u+d_2}{au^3+bu^2+cu+d},\quad 0\leqq u\leqq 1 \qquad (1-1)$$
$$z=\frac{a_3u^3+b_3u^2+c_3u+d_3}{au^3+bu^2+cu+d}$$

（1－1）なる3次の有理関数形であらわされるパラメトリック方程式である。ただしuは助変数で，a_1，a_2，a_3，$a\neq 0$

② Segment 同志の接続条件



SegmentP の$u=1$の点とSegmentQ の$u=0$の点が等しく，ベクトルV_iに等しいこと，SegmentP の$u=1$における接ベクトルV^P_{iu}とSegmentQ の$u=0$における接ベクトルV^Q_{iu}が等しい。さらに$V^P_{iuu}=V^Q_{iuu}$である。すなわち

$$V^P_i=V^Q_i$$
$$V^P_{iu}=V^Q_{iu} \qquad (1-2)$$
$$V^P_{iuu}=V^Q_{iuu}$$

解析

同次座標$wV=[wx,\ wy,\ wz,\ w]$を用いて（1－1）を書き直おすと，$w=au^3+bu^2+cu+d$において，

$$wV=w[x\quad y\quad z\quad 1]$$
$$=[wx\quad wy\quad wz\quad w]$$
$$=[u^3\quad u^2\quad u\quad 1]A \qquad (1-3)$$

ただしAは4×4のマトリックスでSegment を規定するものである。

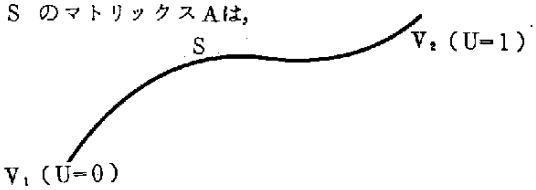
$$A=\begin{bmatrix} a_1 & a_2 & a_3 & a \\ b_1 & b_2 & b_3 & b \\ c_1 & c_2 & c_3 & c \\ d_1 & d_2 & d_3 & d \end{bmatrix} \qquad (1-4)$$

また$(wV)'=[(wx)'\ (wy)'\ (wz)'\ w']$と定義すれば(1－3)から

$$(wV)'=[3u^2\ \ 2u\ \ 1\ \ 0]A \qquad (1-5)$$

$$(wV)''=[6u\ \ 2\ \ 0\ \ 0]A \qquad (1-6)$$

(1－3), (1－5)からSegmentS のマトリックスAは,



$$\begin{bmatrix} w_1 V_1 \\ w_2 V_2 \\ (w_1 V_1)' \\ (w_2 V_2)' \end{bmatrix}=\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} A \qquad (1-7)$$

$$\therefore A=\begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} w_1 & 0 & 0 & 0 \\ 0 & w_2 & 0 & 0 \\ w_1' & 0 & w_1 & 0 \\ 0 & w_2' & 0 & w_2 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_1' \\ V_2' \end{bmatrix} \qquad (1-8)$$

M

ここで$w_1(1)=w_2(1)=\cdots\cdots=w_{n+1}(1)=1$

$w_1'(1)=a_1$, $w_2'(1)=a_2$, ……, $w'_{n-1}(1)=a_{n-1}$, $w'_n(1)=a_n$

という仮定を入れる。

$$A=M\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ a_1 & 0 & 1 & 0 \\ 0 & a_2 & 0 & 0 \end{bmatrix} \qquad (1-9)$$

一般にi番目のSegment のマトリックスA_iは

$$A_i=M\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ a_i & 0 & 1 & 0 \\ 0 & a_{i+1} & 0 & 1 \end{bmatrix} \qquad (1-10)$$

で与えられる。

また②の接続条件から

$$\begin{bmatrix} V_2' \\ V_3' \\ \vdots \\ V_{n-1}' \end{bmatrix} = \begin{bmatrix} 4 & 1 & & & \\ 1 & 4 & 1 & & \\ & 1 & 4 & 1 & \\ & & & & \\ & & & 1 & 4 & 1 \end{bmatrix}^{-1} F \begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_n \end{bmatrix} - \begin{bmatrix} V_1' \\ 0 \\ \vdots \\ 0 \\ V_n' \end{bmatrix} \qquad (1-11)$$

$(n-2)\times 4$　$(n-2)\times(n-2)$　　$n\times 4$　$(n-2)\times 4$

但し

$$F = \begin{bmatrix} -3-a_1 & a_1+a_3 & 3-a_3 & 0 \cdots\cdots & & & \\ 0 & -3-a_2 & a_2+a_3 & 3-a_4 & & & \\ 0 & \vdots & & & & & \\ \vdots & \vdots & & & & & \\ 0 & 0 & & & & & \\ 0 & 0 & \cdots\cdots & & -3-a_{n-2} & a_{n-2}+a_n & 3-a_n \end{bmatrix}$$

$(n-2)\times n$

Fitting のアルゴリズム

データ；通過点の同次座標 V_1，V_2，……V_n

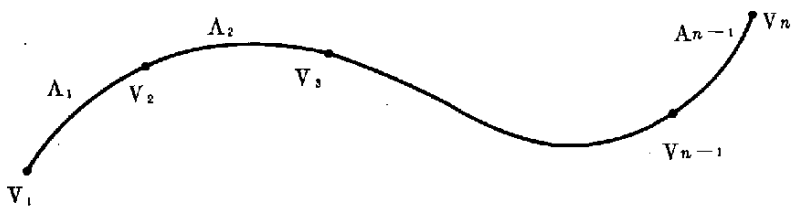
両端点の接ベクトル V_1'，V_n'

パラメータ a_1，a_2，……a_n

STEP1　接ベクトル V_2'，V_3'，……，V_{n-1}' を（1－11）によって計算する。

STEP2　各Segment の方程式を決定するマトリックスAを（1－10）で計算する。

以上ですべてのSegment のマトリックスが決まるので，全曲線が決ったことになる。

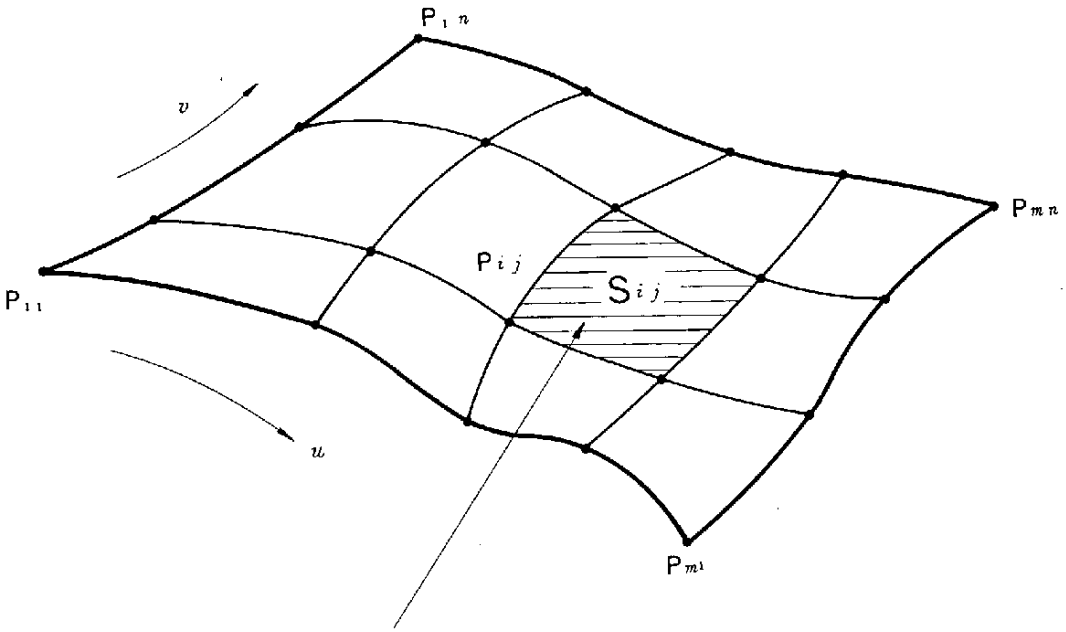


パラメータ a_1，a_2，……a_n；a_1，a_2，……，a_n を勝手に選んでも，前の方法に従えば各点 V_1，V_2，……，V_n をFitting することは可能である。ところが a_1，……a_n を変化させると固定点 V_1，V_2，……，V_n 以外の点がさまざまに動くので，パラメータの変化と各Segment の動きとの関係がわかると，パラメータをうごかすことによって曲線を修正できる。

(III)　Surface fitting

本方法は下図の様に $m\times n$ の格子状に分布した点群を通って C^2 級のなめらかさで曲線を創成す

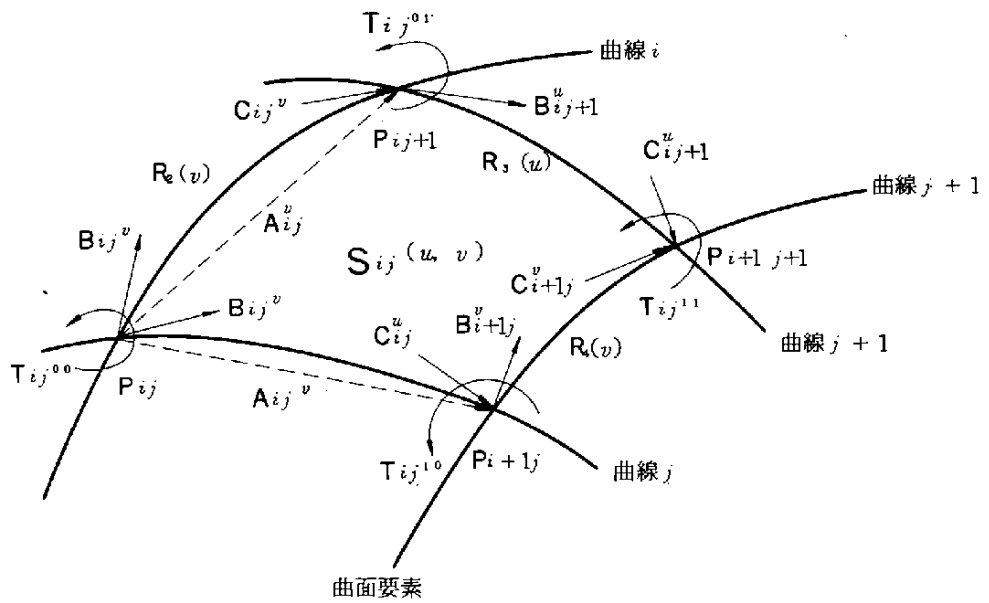
るものである。



（Sij）のSurface Segmentまたは Patch

本方法の骨子は，まず曲面要素（Patch）を穂坂の式で補間し，次に全曲面がC^2級のなめらかさになる様に各Patchにおける補間式のパラメータを初期データをもとにして，連立一次方程式で求めるものである。

解析



① 上図の曲面要素を穂坂の式を用いて書き下すと，

$$\begin{aligned} S_{ij}(u,\ v) &= R_1(u) + \{R_3(u) - R_1(u)\} f(v) \\ &\quad + R_2(v) + \{R_4(v) - R_2(v)\} f(u) \\ &\quad + \frac{1}{2}\{R_{11} + R_{21} - R_{31} - R_{41}\} f(u) f(v) \\ &\quad + T_{ij}^{00} g(u) g(v) + T_{ij}^{01} g(u) h(v) \\ &\quad + T_{ij}^{10} h(u) g(v) + T_{ij}^{11} h(u) h(v) \\ &\quad + P_{ij} \qquad \cdots\cdots (1) \end{aligned}$$

ただし $R_k(0) = 0,\ R_k(1) = R_{k1}$

$T_{ij}^{kl} = [\partial S_{ij}(u,\ v)/\partial u \partial v]_{u=k,\ v=l} \qquad k,\ l = 0,\ 1$

$0 \leqq u \leqq 1,\quad 0 \leqq v \leqq 1$

② 曲面要素の図で各ベクトルを定義する。

$$\left.\frac{\partial S_{ij}}{\partial u}\right|_{u=0,\ v=0} = B_{ij}^{u} \qquad \left.\frac{\partial S_{ij}}{\partial u}\right|_{u=1,\ v=0} = C_{i,j}^{u}$$

$$\left.\frac{\partial S_{ij}}{\partial v}\right|_{u=0,\ v=0} = B_{ij}^{v} \qquad \left.\frac{\partial S_{ij}}{\partial v}\right|_{u=0,\ v=1} = C_{ij+1}^{v}$$

$$\left.\frac{\partial S_{ij}}{\partial v}\right|_{u=0,\ v=0} = B_{i,\ j+1}^{u} \qquad \left.\frac{\partial S_{ij}}{\partial v}\right|_{u=1,\ v=1} = C_{ij+1}^{v}$$

$$\left.\frac{\partial S_{ij}}{\partial v}\right|_{u=1,\ v=0} = B_{i+1,\ j}^{v} \qquad \left.\frac{\partial S_{ij}}{\partial v}\right|_{u=1,\ v=1} = C_{i+1,\ j}^{v}$$

$$\left.\frac{\partial^2 S_{ij}}{\partial u \partial v}\right|_{u=0,\ v=0} = T_{ij}^{00} \qquad \left.\frac{\partial^2 S_{ij}}{\partial u \partial v}\right|_{u=1,\ v=0} = T_{ij}^{10}$$

$$\left.\frac{\partial^2 S_{ij}}{\partial u \partial v}\right|_{u=0,\ v=1} = T_{ij}^{01} \qquad \left.\frac{\partial^2 S_{ij}}{\partial u \partial v}\right|_{u=1,\ v=1} = T_{ij}^{11}$$

$$A_{ij}^{u} = P_{i+1,\ j} - P_{ij} \qquad A_{ij}^{v} = P_{i,\ j+1} - P_{ij}$$

$$A_{ij+1}^{u} = P_{i+1,\ j+1} - P_{i,\ j+1} \qquad A_{i+1,\ j}^{v} = P_{i+1,\ j+1} - P_{i+1,\ j}$$

$$k_{i+1,\ j}^{u} = |A_{ij}^{u}| / |A_{i+1,\ j}^{u}|, \qquad k_{i,\ j+1}^{v} = |A_{ij}^{v}| / |A_{ij+1}^{v}|$$

③ 上で定義したベクトル量を用いて(1)を書きかえる。

いま曲面要素の図で4本の境界線 $R_1(u)$, $R_2(v)$, $R_3(u)$, $R_4(v)$ をそれぞれの弦ベクトルおよび両端点の接線ベクトルを用いて

$$\left.\begin{aligned} R_1(u) &= A_{ij}^{u} f(u) + B_{ij}^{u} g(u) + C_{ij}^{u} h(u) + D_{ij}^{u}(u) \\ R_3(u) &= A_{i\,j+1}^{u} f(u) + B_{i\,j+1}^{u} g(u) + C_{i\,j+1}^{u} h(u) + D_{i\,j+1}^{u}(u) \\ R_2(v) &= A_{ij}^{v} f(v) + B_{ij}^{v} g(v) + C_{ij}^{v} h(v) + D_{ij}^{v}(v) \\ R_4(v) &= A_{i+1,j}^{v} f(v) + B_{i+1,\ j}^{v} g(v) + C_{i+1,j}^{v} h(v) + D_{i+1,\ j}^{v}(v) \end{aligned}\right\} \qquad (2)$$

と仮定できる。

ここで境界条件

$R_i(0)=0$, $R_i(1)=A_i$

$\dot{R}_i(0)=B_i$, $\dot{R}_i(1)=C_i$

を代入し，且つ$f(t)$, $g(t)$, $h(t)$($t-u$, v)をtの多項式と仮定すればD^u_{ij}, $D^u_{i\,j+1}$, D^v_{ij}, $D^v_{i+1\,j}=0$ および

$f(t)=-2t^3+3t^2$

$g(t)=t^3-2t^2+t$

$h(t)=t^3-t^2$

となる。これを(2)式に代入すれば，

$$\left.\begin{aligned}
R_1(u)&=A^u_{ij}f(u)+B^u_{ij}g(u)+C^u_{ij}h(u)\\
R_3(u)&=A^u_{ij+1}f(u)+B^u_{ij+1}g(u)+C^u_{ij+1}h(u)\\
R_2(v)&=A^v_{ij}f(v)+B^v_{ij}g(v)+C^v_{ij}h(v)\\
R_4(v)&=A^v_{i+1\,j}f(v)+B^v_{i+1\,j}g(v)+C^v_{i+1\,j}h(v)
\end{aligned}\right\}\quad (2)'$$

$R_{11}=R_1(1)$, $R_{21}=R_2(1)$, $R_{31}=R_3(1)$, $R_{41}=R_4(1)$であるから

$R_{11}=A^u_{ij}$, $R_{21}=A^v_{ij}$, $R_{31}=A^u_{ij+1}$, $R_{41}=A^v_{i+1\,j}$ が成立する。

以上の結果から(1)は次の様になる。

$$\begin{aligned}
S_{ij}&(u,v)\\
&=\{A^u_{ij}f(u)+B^u_{ij}g(u)+C^u_{ij}h(u)\}\{1-f(v)\}\\
&+\{A^u_{ij+1}f(u)+B^u_{ij+1}g(u)+C^u_{ij+1}h(u)\}\cdot f(v)\\
&+\{A^v_{ij}f(v)+B^v_{ij+1}g(v)+C^v_{ij}h(v)\}\{1-f(u)\}\\
&+\{A^v_{ij+1}f(v)+B^v_{ij+1}g(v)+C^v_{ij+1}h(v)\}f(u)\\
&+\frac{1}{2}\{A^u_{ij}+A^v_{ij}-A^u_{ij+1}-A^v_{i+1\,j}\}f(u)f(v)\\
&+T^{00}_{ij}g(u)g(v)+T^{01}_{ij}g(u)h(v)+T^{10}_{ij}h(u)g(v)+T^{11}_{ij}h(u)h(v)+P_{ij}
\end{aligned}\qquad\cdots\cdots(1)'$$

ただし$t=u$, vとすると

$f(t)=-2t^3+3t^2$

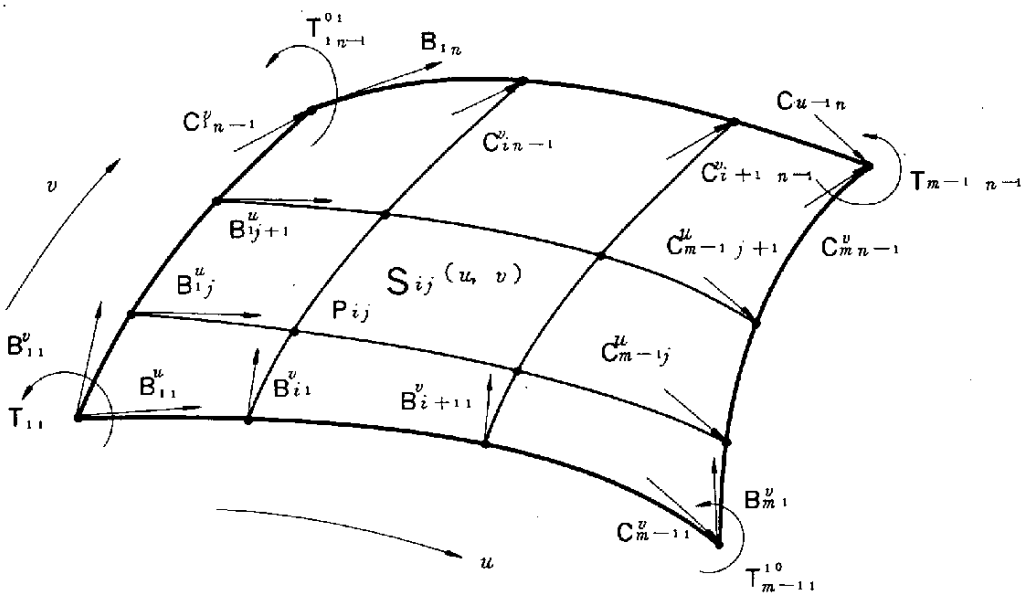
$g(t)=t^3-2t^2+t$

$h(t)=t^3-t^2$

式(1)′を見ると，弦ベクトルA^u_{ij}, A^u_{ij+1}, A^v_{ij}, A^v_{ij+1} ； 接ベクトルB^u_{ij}, B^u_{ij+1},

B^v_{ij}, $B^v_{i+1\ j}$, C^u_{ij}, C^u_{ij+1}, C^v_{ij}, C^v_{ij+1} ；偏微分係数 T^{00}_{ij}, T^{01}_{ij}, T^{01}_{ij}, T^{11}_{ij} が与えられれば（ $i\ j$ ）のPatch上の点 S_{ij}（ u, v ）は求められることがわかる。これらの定ベクトルは先にのべたPatchのパラメータであって，これらのパラメータと初期データとの関係を以下に述べる。

④ 初期データと各Patchのパラメータとの関係



<初期データ> P_{ij} ； $i=1,\ \cdots\cdots m,\ \ j=1,\ \cdots\cdots,\ n$（点の位置ベクトル）

B^u_{1j}, $C^u_{m-1\ j}$ ； $j=1,\ \cdots\cdots,\ n$

（曲面全体のへりに存在する格子点における u 曲線に沿う勾配ベクトル）

$B^v_{i\,1}$, $C^v_{i,\ n-1}$ ； $i=1,\ \cdots\cdots,\ m$

（曲面全体のへりに存在する格子点における v 曲線に沿う勾配ベクトル）

T^{00}_{11}, $T^{10}_{m-1\ \ 1}$, $T^{01}_{1\ n-1}$, $T^{11}_{m-1\ n-1}$

（曲面全体の4隅の偏微分ベクトル）

以上（ $mn+2n+2m+4$ ）=（ $m+2$ ）（ $n+2$ ）個の定ベクトルであるが，これらと各Patchのパラメータとの関係は次の様である。

$$B^u_{ij}+2k^u_{i+1\ \ j}(1+k^u_{i+1\ \ j})B^u_{i+1\ \ j}+(k^u_{i+2\ \ j}\ B^u_{i+2\ \ j})$$
$$=3\left\{A^u_{ij}+(k^u_{i+1\ \ j})^2 A^u_{i+1\ \ j}\right\} \qquad (3)$$
$$i=1,\ \cdots\cdots,\ m-2$$

$$B^v_{ij} + 2\,k^v_{i\ j+1}\,(1+k^v_{ij+1})B^v_{ij+1} + (k^v_{ij+1})^2\,k^v_{ij+2}\,B^v_{ij+2}$$
$$= 3\left\{A^v_{ij} + (k^v_{ij+1})^2\,A^v_{ij+1}\right\} \tag{4}$$
$$j = 1,\ \cdots\cdots,\ n-2$$

(3), (4)はu曲線, v曲線のC^2級の連続条件から出て来たものである。

$$T^{00}_{ij} + 2\,k^u_{i+1\ \ j}\left\{1+k^u_{i+1\ \ j}\right\}T^{00}_{i+1\ \ j} + (k^u_{i+1\ \ j})^2\,k^u_{i+2\ \ j}\,T^{00}_{i+2\ \ j}$$
$$= 3\left\{B^v_{i+1\ \ j} - B^v_{i,\ j} + (k^u_{i+1\ \ j})^2\,(B^v_{i+2} - B^v_{i+1,\ j})\right\} \tag{5}$$
$$i = 1,\ \cdots\cdots,\ m-2\ ;\ j = 1,\ \cdots\cdots n$$

$$T^{00}_{i\ j} + 2k^v_{i\ j+1}\left\{1+k^v_{ij+1}\right\}T^{00}_{ij+1} + (k^v_{i\ \ j+1})^2\,k^v_{ij+2}\,T^{00}_{ij+2}$$
$$= 3\left\{B^u_{ij+1} - B^u_{ij} + (k^v_{i\ \ j+1})^2\,(B^u_{ij+2} - B^u_{ij+1})\right\} \tag{6}$$
$$j = 1,\ \cdots\cdots,\ n-2\ ;\ i = 1,\ \cdots\cdots m$$

(5), (6)はそれぞれ

$$\left.\frac{\partial^3 S_{ij}(u,\ v)}{\partial u^2\,\partial v}\right|_{\substack{u=1\\v=0}} = (k^u_{i+1\ \ j})^2 \left.\frac{\partial^3 S_{i+1\ \ j}(u,\ v)}{\partial u^2\,\partial v}\right|_{\substack{u=0\\v=0}}$$
$$\left.\frac{\partial^3 S_{ij}(u,\ v)}{\partial u\,\partial v^2}\right|_{\substack{u=0\\v=1}} = (k^v_{ij+1})^2 \left.\frac{\partial^3 S_{ij+1}(u,\ v)}{\partial u\,\partial v^2}\right|_{\substack{u=0\\v=0}}$$

なる条件から得られる。

また

$$C^u_{ij} = k^u_{i+1\ \ j}B^u_{i+1\ j} \qquad i = 1,\ \cdots\cdots,\ m-1\ ;\ j = 1,\ \cdots\cdots,\ n$$
$$C^v_{ij} = k^v_{ij+1}\,B^v_{ij+1} \qquad i = 1,\ \cdots\cdots,\ m\ ;\ j = 1,\ \cdots\cdots n-1$$
$$T^{10}_{ij} = k^u_{i+1\ \ j}T^{00}_{i+1\ j} \qquad i = 1,\ \cdots\cdots;\ m-1\ ;\ j = 1,\ \cdots\cdots n \tag{7}$$
$$T^{01}_{ij} = k^v_{ij+1}\,T^{00}_{ij+1} \qquad i = 1,\ \cdots\cdots,\ m\ ;\ j = 1,\ \cdots\cdots n-1$$
$$T^{11}_{ij} = k^u_{i+1\ \ j+1}\,k^v_{i+1\ \ j+1}\,T^{00}_{i+1\ \ j+1} \qquad i = 1,\ \cdots\cdots,\ m-1\ ;\ j = 1,\ \cdots\cdots n-1$$

が成立する。

アルゴリズム

$$\left.\begin{aligned} A^u_{mj} &= A^u_{m-1} = P_{mj} - P_{m-1\ j}\\ A^v_{in} &= A^v_{in-1} = P_{in} - P_{in-1}\end{aligned}\right\}\ \text{なる仮定を導入する。}$$

これは，$B^{u}_{mj}=C^{u}_{m-1\ j}$，$B^{v}_{in}=C^{v}_{i\ n-1}$，$T^{10}_{m-1\ \ 1}=T^{00}_{m\ 1}$，$T^{11}_{m-1\ \ n-1}=T^{00}_{m\ n}$ と仮定したのと同じである。

先ほどの初期データとこの仮定をもとにして各Patchのパラメータを(3)，(4)，(5)，(6)，(7)によって求める。それを(1)′に代入すれば各Patchが創成できたことになる。

3.4 隠れ線消去 (Hidden line Elimination)

隠れ線消去の問題は，陰影問題とともに作図において重要なものである。Compnter Graphics におけるこの問題は多面体の場合からスタートとした。多面体の場合は，図形が平面で構成されていること，およびその投象が多角形（polygon）になることなどの理由で，線形代数初等的なベクトル解析などを使うことによって処理できる。この多面体の隠れ線消去の具体的な方法は後述するとして，一般に複雑な多面体や曲面の場合は，従来行われて来た上記のアルゴリズムでは処理時間に難点があり別の方法が考えられている。これについては参考文献を参照されたい。また，一般的な曲面でなく特別な曲面，たとえば2次曲面や，回転面などはその性質をうまく利用することによってこの種の問題を上手に解決できるのではないかと思われるが，あまり研究されていない。

さて次に多面体の隠れ線消去の一手法を紹介するわけであるが，ここで紹介するものは，R. Galimbert and U. Montanari, "An Algolitthm for Hidden Line Elimination", Communications of the ACM, vol. 12/No. 4/April, 1969である。

定義

① edge；面face の交線

② node；edge の両端点

③ edgeが凸状の交線ならばそのedgeは"convex"である。凹状の2面の交線ならばそのedgeは"concave"であるという。

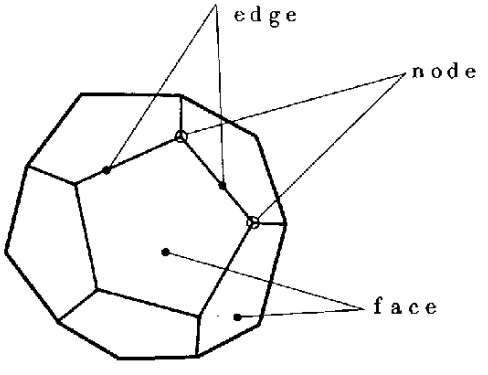


図1 凸多面体

図1ではすべてのedgeはconvex である。図2において j_1 と j_2 の交線となっているedgeはconcave の例である。

また図3の j_1 と j_2 の交線であるedgeはconvex の例である。

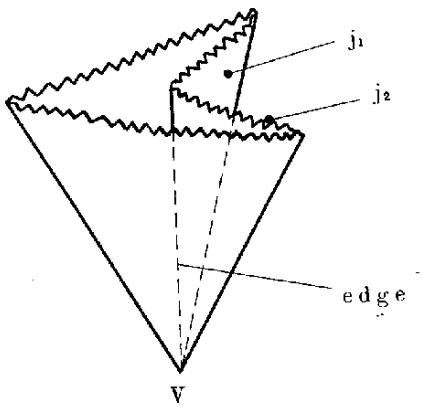


図2

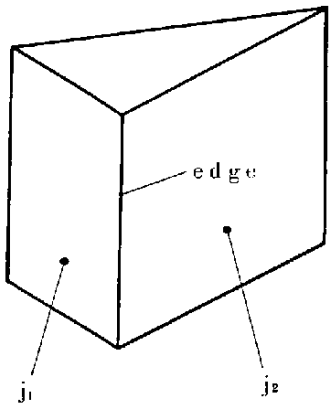


図3

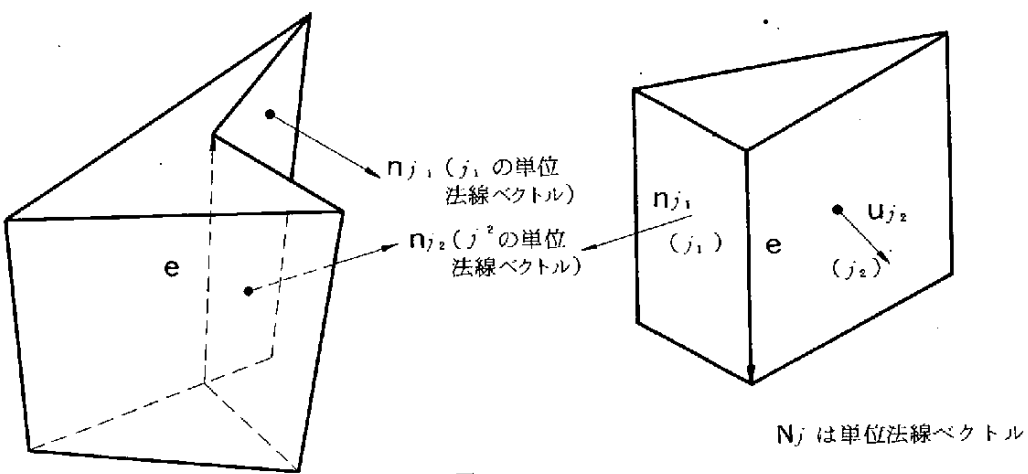
④ nodeが少くとも1本のconcave edgeの端点であるならば，このnodeは"concave"と呼ばれる。そうでないときは"convex"と言う。図1ではすべてのnadeはconvex である。図2ではVは，1本のconcave edgeの端点になっているので，concave である。図3は図1と同様にすべてのnodeはconvexである。

⑤ ①，②で定義したedge，node，face というのは，3次元における空間図形の名称であるが，これをpicture planeに投象したとき次の様に名称を変えて区別する。

3次元（原像）	投　象	2次元（写像）
faces		polygons
edges		segments
nodes		vertices

edgeの評価（STEP1）

edgeがcenvexかconcaveかという区別をどの様に数量化するかという事を述べる。



N_j は単位法線ベクトル

図　4

$C = e \cdot (n_{j_1} \times n_{j_2})$なるベクトルの内積をとることによってedgeは評価される。

$C > 0$; edge eはconcave
$C < 0$; edge eはconvex
} である。

但し，eの方向はfacej_1を右に見る方向である。もし，eがfacej_2を右に見る方向にとるならば評価式は$C = e \cdot (n_{j_2} \times n_{j_1})$となる。

nadeの評価は，それを端点に有するedgeを調べればconvexかconcaveかが決められる。かくして3次元物体の凹凸の状態を把握できる。

すなわち各edgeのconvex, concave, 各nadeのconvex, concave, 各faceの単位法線ベクトルを知ることが出来る。この3つのことを知ることがアルゴリズムの第一段階である。

見える面，見えない面の評価（ＳＴＥＰ２）

図5に示す様に視点をE，Eの方向に向いたpicture planeの垂直ベクトルをV，n_{ji}をfacej_iの法線ベクトルとすれば

$C' = V \cdot n_{j_i}$

に於て

$C' > 0$; facej_iは「見える可能性のある面」

$C' \leq 0$; facej_iは「見えない面」

である。

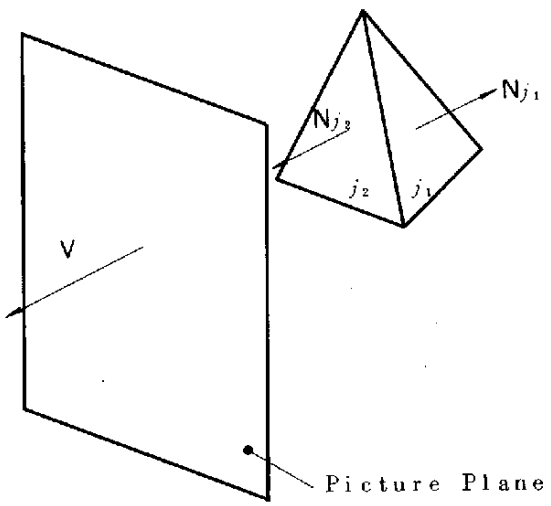


図　5

アルゴリズムの第2段階は各面の性質"見える可能性のある面"と見えない面を$C' = V \cdot n_{j_i}$の評価式によって区別することである。そして次の規則によって，かくれ線消去の第1段が行われる。

(a) 2つの見えない面（hidden face）に属するedgeの投象（これをsegmentと言うが）は消去すること。

(b) edgeがconcaveで，それを共有している2面が，一方は「見えない面」であり，一方は「見える可能性のある面」である場合，そのsegmentは消去する。

これを実例で説明しよう。図6の図形の各edge, node, faceの性質は次頁の様になる。

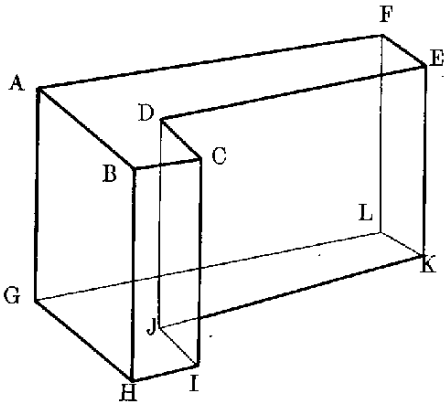


図 6

edge	convex	concave
AB	○	
BC	○	
CD	○	
DE	○	
EF	○	
FA	○	
AG	○	
BH	○	
CI	○	
DJ		○
EK	○	
FL	○	
GH	○	
HI	○	
IJ	○	
JK	○	
KL	○	
LG	○	

node	convex	concave
A	○	
B	○	
C	○	
D		○
E	○	
F	○	
G	○	
H	○	
I	○	
J		○
K	○	
L	○	

face	見えない面C′ ≤0	見える可能性のある面C′ >0
ABCDEF		○
GHIJKL	○	
AGHB		○
BHIC		○
CIJD	○	
DJKE		○
EKLF	○	
FAGL	○	

規則(a)によって消去されるsegment

GL；見えない面AGLF，GHIJKLに属するから

FL；見えない面AGLF，FLKEに属するから

LK；見えない面FLKE，GHIJKLに属するから

JI；見えない面GHIJKL，DJICに属するから

規則(b)によって消去されるsegment

DJ；見える可能性の面DJKE，見えない面DJICに属する，concaveであるから

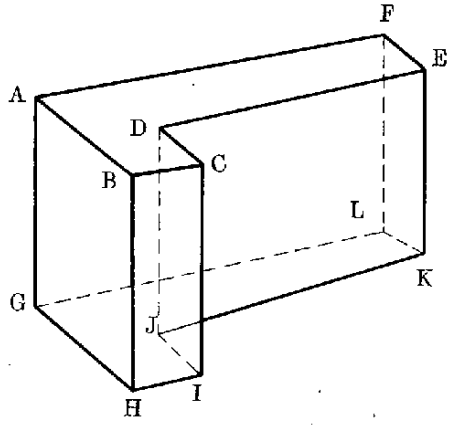


図 7

あとJKの一部を消去すれば完全にかくれ線消去が完成するが，ＳＴＥＰ２までの判定ではそれはできない。このように残った未だ消されていないsegment については次に述べるＳＴＥＰ３で消すことが出来る。もし物体のすべてのedgeがconvex ならばＳＴＥＰ２でかくれ線の消去がすべて完了する。

segment 上の点のnature の決定

図７において，segmentＪＫ を見ていただきたい。ＪＫはsegmentＣＩ と交わっているが，その交点をＱとすれば，ＪＱはかくれ線であるので消去し，ＱＫはそのままにしておきたい。それにはＪＫ上の各点のnature を決定しなければならない。このアルゴリズムを述べるのがここの目的である。

アルゴリズムを述べる前にいくつかの簡単な定義を述べる。

① 図７においてsegmentＣＩはsegmentＪＫの上になっているので，交点Ｑはsegment ＣＩに関して"winning"と言い，segmentＪＫ に関して交点Ｑは"losing"と言う。

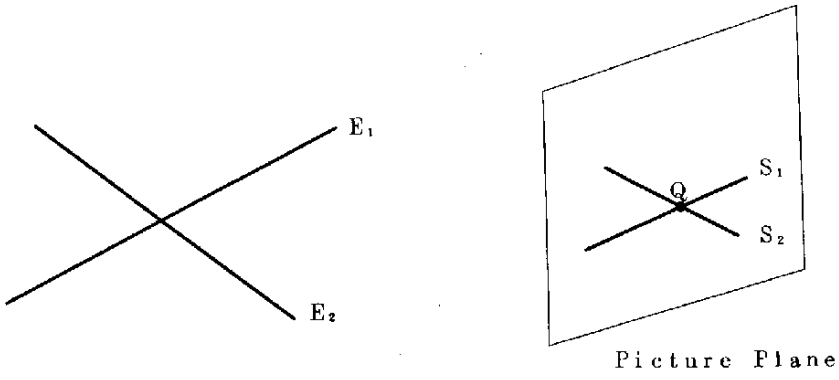
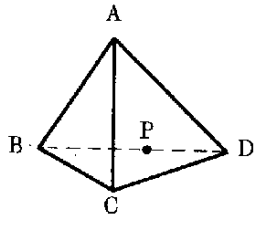


図 8

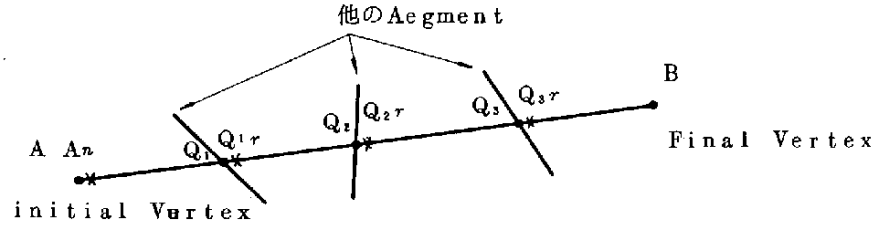
一般に，edge e_1 がedge e_2 の上になっているとき，picture planeにうつし出されたsegment S_1 とsegment S_2（S_1 はe_1 の投象，S_2 はe_2 の投象）の交点QはS_1 に関してwinning，S_2 に関してlosingと言う。

またP点を一つのsegment S 上の点であるとする。Pの原像P′（もちろんSの原像であるedge e 上にP′点は存在する）を隠すon−view faceの集合をP点の"nature"Hと呼ぶ。

（例） BD上のP点のnature Hは{△ACD}である。



次に，一本のsegment 上の点のnature を決定する方法を述べる。



検定しようとするSegment

図 9

図9の様に検定しようとするsegment とこれに交わる他のsegment が存在する。検定しようとするsegment の端点の一方をinitial vertex，他方をfinal vertexと呼ぶ。これはsegment 上の点のnature の決定順序によってきまるものである。図9では左から右へ各

点のnatureを決定する場合である。

natureの決定は，segmentのすべての点について調べるわけでなく，initial vertex A点，Aに無限に近い点A_n，他のsegmentとの交点Q_iに無限に近い点Q_{ir}，final vertex B点である。これはAQ_1，Q_1Q_2，……上のすべての点はA_n，Q_{1r}，……のnatureと同一であるという理由にもとづく。

(i) initial vertexのnature H の決定

(イ) initial vertex V_i がon-view face j の投象であるpolygon j に属すること。

(ロ) 視点をP_0，V_iの原像をP_iとすれば$\overline{P_0P_i}$がface j を貫いていること。

(イ)，(ロ)を同時に満すときV_iのnatureはpolygon j である。

(ii) initial vertexに無限に近い点のnature H_n の決定

Case a；initial vertexが凸(convex)で，かつlosing intersectionでないとき。

$$H_n = H$$

Case b；initial vertexが凹(concave)で，かつlosing intersectionでない場合。

（例） segment JKにおいてinitial vertex Jは凹である。しかもDJとJKが実際に交わっているからJはcase b の条件に合う。

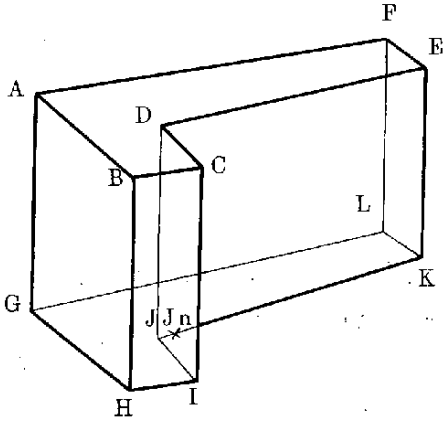
$$H_n = H + P' \quad (P' \subset P)$$

$$H = H_n - (H_n \cap P)$$

H；initial vertexのnature

P；on-view faceの集合

点Jがその中に属している。



Case c；initial vertexが凸で，かつlosing intersectionである場合。

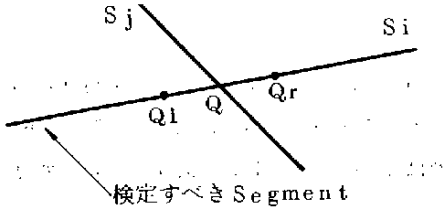
$$H_n = H - P_n \quad (P_n \subset R)$$

R；S_jを共有するon-view faceの集合

Case d；initial vertexが2つ以上のlosing intersectionかまたは凹であるならば，すべての対応する集合RおよびPを考察することによって遂行される。

(iii) segment の交点におけるnature

Qに無限に近い左側の点をQ_l，右側の点をQ_rとする。Q_l，Q_rのnatureをそれぞれH_l，H_rとしたときH_rを求める。H_lは既知



であることに注意する。

Case a；QがS_iに関してwinningの場合

$$H_r = H_\ell$$

Case b；QがlosingでS_jの端点でないとき

（b－1） 集合Rが1つの元しか含まない場合（R；S_jを共有するon-view faceの場合）

$$\begin{cases} R \subset H_\ell \text{ ならば } H_r = H_\ell - R \\ R \not\subset H_\ell \text{ ならば } H_r = H_\ell + R \end{cases}$$

（b－2） 集合Rが2つのface j_ℓ, j_rから成立しているとき，$R=\{j_\ell, j_r\}$ j_ℓがH_ℓに属している

$$H_r=(H_\ell-\{j_\ell\})+\{j_r\}$$

Case c；QがlosingでS_jの端点であるとき

（C－1） S_iとS_jが重なっているとき，

Q_ℓからQ_rへ時計方向にまわることによってS_jが重ならないとき

$$H_r=H_\ell$$

そうでないとき

$$H_r=H_\ell+R \quad \text{or} \quad H_r=(H_\ell-\{j_\ell\})+\{j_r\}$$

（c－2） S_iとS_jが重なっているとき

$R \subset H_\ell$ ならば $H_r=H_\ell-R$

$R \not\subset H_\ell$ ならば $H_r=H_\ell \cup R$

Case d；もしQ点において，2つ以上の交点が重なっているとき，natureの変化はすべての交点の組み合せで行う。

(Ⅳ) final vertexのnature Hの決定

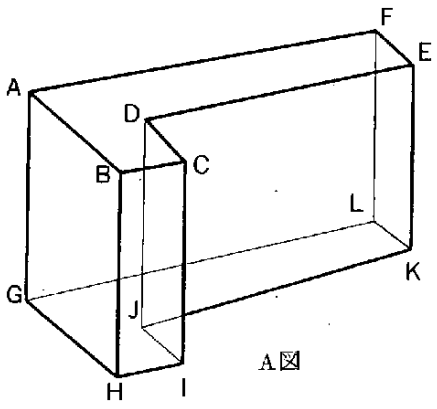
final vertex に無限に近い点のnature H_n はいままでの議論で既知であるから(Ⅲ)における3つの公式を適用すればHは求められる。

例題

STEP1，STEP2，STEP3を順次適用して，かくれ線消去の過程を見る。

右図の様な逆L形の多面体を透視変換によって出力を行う。このときいわゆるwire stractureとなっている。この多面体について，各edgeの性質，各nodeの性質，各面のnormal vectorを求めSTEP2の操作を行うとB図になる。

segment CI とsegment JK との交点をQとする。JKは未だ消されずに残っているsegme－



A図

⇩

nt である。ＪＫについてＳＴＥＰ３を適用しよう。

Ｊのｎａｔｕｒｅ ；□ＢＨＩＣ

J_n のｎａｔｕｒｅ ；(ii)のＣａｓｅｂ を適用して

□ＢＨＩＣ

Q_ℓ のｎａｔｕｒｅ ；H_ℓ＝{□BHIC}

Ｃ･Ｉを有するｏｎ－ｖｉｅｗ ｆａｃｅの集合をＲとすれば

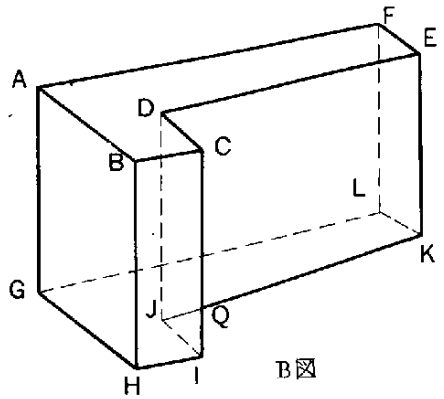
Ｒ＝{□ＢＨＩＣ}

点Ｊはｃｏｎｃａｖｅ

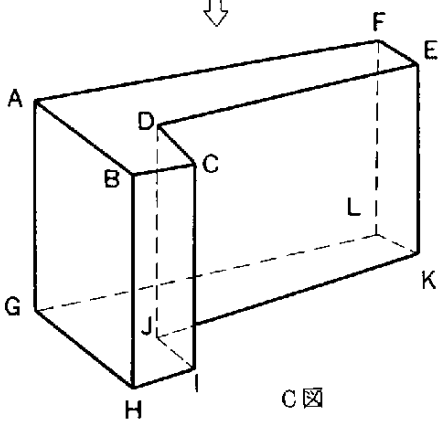
故に（ｂ－１）から$R \subset H_\ell$ だから

$H_r = H_\ell - R = \phi$

よってＪＱはかくれ線とわかり消去，ＱＫはそのままと判定できる。こうして得られたのがＣ図である。



Ｂ図



Ｃ図

3.5 参考文献

Computer Graphicsにおける数学もComputer Graphicsという全体と，数学特に幾何学全体とを背景に考える必要がある。これに関しては，たとえば

(1) The Computer Display Review

(2) 沖野教郎，"コンピュータによる自動デザイン"，日刊工業新聞社，1967

(3) 長野正，"曲面の数学"，培風館，1968

(4) D. V. Ahuja and S. A. Coons, "Geometry for construction and display", IBM System Journal vol 7/Number 3&4, P. 188~205

など見られたい。(1), (2)はComputer Graphics全体の中での数学を，(3)は幾何学全体を見わたすのによいものである。(4)は同次座標，投象問題，曲線，曲面のFitting などの様子を知るのに役立つ。始めに述べた様にComputer Graphicsにおける数学は，線形代数，座標幾何，テンリル解析，ベクトル解析などを手段に用いるのであるから，この方面の参考書も見られたい。

また，第1～4節で述べた手法に関連するものは次の通りである。

PICTURE CALCUS に関するもの

(5) W. F. Miller and Alan C. Shaw, "A Picture Calculus"

(6) Alan C. Shaw, "The Formal Description and Parsing of Pictures"

(7) Alan C. Shaw, "On the interactive Generation and interpretation of Artifical Pictures"

投象問題に関するもの

基礎的なものとしては(4)があげられるが，同次座標の概念については(3)の4章の射影平面を見られたい。透視マトリックスなどのまとめとしては

(8) B. Herzog, "Some Useful Geometry for Computer Graphics"

がある。またステレオグラフなどの話題を盛ったものとしては

(9) A. J. Cole, "Plane and stereographic projection of convex polybedra from minimal information"

(10) D. Ophir, B. J. Shepherd, and R. J. Spinrod, "Three-Dimentional Computer Display", Communication of the ACM

などがある。

Fitting, Smoothingに関するもの

もともとフィッテングやスムーシングの技術は自由曲面(Free-Form Surface)の創成に始まる。その工学的背景としては

(11) 日本電子工業振興協会，"機械設計のためのプログラムの研究"，１９６８

(12) 機械振興協会技術研究所，"自動設計に関する調査"．１９６８

(13) 研野和人，"数値制御工作機械の自動プログラミング（曲面創成と加工用プログラム）について" 日本機械学会誌，第７１巻，第５９０号，１９６７

などがある。

またFitting Smoothingの理論およびアルゴリズムについては，すでに記した(11)の他に

(14) D. V. Ahja, "An algorithm for generating spline-like curves" IBM system Journal, volume Seven/Numbers Thres & Four/1968 P. 206～217

(15) 穂坂衛，"曲線，曲面の合成および平滑化理論"，情報処理，vol. 10, No. 3, 1969

(16) 東芳宏，兵藤良博，高崎直樹，"数値制御三次元プログラム（ＨＡＰＴ－３Ｄ）について"，日本機械学会誌，vol. 71, No. 590

(17) Theodere M. P. Lee, "A class of surface for computer display" SJCC, 1969

(18) William J. Gordon, "Free-Form Surface interpolation through curve networks", General Motors Reserch Laboratories, September 30, 1969

(19) Dan Cohen and Fheodore M. P. Lee, "Fast drawing of curves for computer display", SJCC, 1969

(20) D. G. Hayes, "A method of storing the orthogonal polynominals used for curve and surface fitting"

(21) 岡田吉誼，"曲線のフェアリングに関する一実験"，情報処理，vol. 10, No. 3, 1969

(22) 藤野孝爾，"曲面のフィッテングとスムージング，Journal of the J. S. M. E, vol. 71, No. 590

(23) S. A. Coons and B. Herzog, "Surfaces for Computer-Aided Aircraft Design"

(24) 平井幸雄，亀多正人，"Toyo Automatic Drawing-2D imensional subroutine package"

などがある。(11), (14)は第３節で紹介を行ったが，いわゆる穂坂の式は(15)に詳しい。(16)は別のＦｉｔｔｉｎｇ方法のアルゴリズムを示し，(21)は(11), (15), (22)とともにSmoothingの方法を述べている。(17)については曲面フィッテングのガイダンスであり，いろいろな話題が述べられている。(18), (19), (20)はフィッテングの手法，(24)は東洋工業株式会社ＴＡＤ－２Ｄについての紹介がそれぞれ述べられている。

隠れ線消去・陰影問題に関するもの

２次元の図形の相互関係の処理を扱ったものとして

(25) J. D. Jacobsen, "Geometric relationships for retrieval of

geographics information", IBM System Journal vol. 7/Numbers 3&4, P. 331~341

(26) Herbert Freeman, "An Algorithm for the Solution of the Two-Dimmensional of "Hidden-Line"Problem", IEEE Transaction of Electronic Computers, vol. EC-16, No. 6, December 1967

などがあるが，第5節で紹介したものは

(27) R. Galimbeti and U. Mantanari, "An Algorithm for Hidden Line Elimination", Communication of the ACM vol. 12/No. 4/April, 1969

である。思想的に同じ系統を引くものとして

(28) 武田智雄，"作図におけるコンピュータの利用"

(29) Arthur Appel, "Some techniques for shading machine renderings of solids", SJCC, 1968

などがある。また理論的な研究論文として

(30) 萩原宏，上内秀隆，"3次元"hidden-line"問題の基礎的考察"，電気通信学会研究資料 A48, 68, IT49(1969-01-21)

がある。

またhidden line 問題の別のアプローチとしては

(31) Chris Wylie, Gordon Romney, David Evans and Alan Eddahl, "Half-tone perspective drawings by computer", FJCC, 1967

(32) John E. Warnork, "The Hidden Line problem and the Use of Halftone Displays"

などがある。

4．三次元図形操作プログラム

4.1 目　　的

我々が現在進めてきている，インタラクティブ・コンピュータ・グラフィックス・プロジェクトの一環として，今年度は次の様な目的のプログラムを実験的に試作した。

(1) 現在設置されているＨＩＴＡＣ－８８１１グラフィック・システムのハードウェア，ソフトウェア両面の能力を実験的に把握し，問題点を探る。

(2) 図形操作の基礎実験を行い，図形処理に慣れ親しみ，その経験を本プロジェクトに生かす。

このプログラムは，三次元図形データ（点と線分の情報）を入力し，このデータに対し移動，回転，拡大縮少等の操作をダイアル，ファンクションキー等を用いて行うものである。

4.2 使用機器構成

使用したシステムはグラフィック専用の小型システムで下記のような構成になっている。

H－８８１１　処理装置
　　１６KW （１６bit/word）
H－８８１２　ディスプレイ制御装置
H－８８１３　グラフィック・ＣＲＴ
　　ライトペン，ファンクションキー，ダイアル付
高速紙テープ読取機
問合せタイプライタ
　　紙テープ・リーダ，パンチ付
ＸＹプロッタ
I/O チャネル　２
H－８８１４　D×C （データ交換装置）
　　（将来，中央の大型システムと結合するため）

このグラフィック・システムは現在紙テープ・ベースとなっており，本プロジェクトの三次元図形操作プログラムの開発に利用したソフトウェア・システムは下記の通りである。

H－８８１１　ＥＸＥＣ （制御プログラム）
H－８８１１　Ａｓｓｅｍｂｌｅｒ （紙テープ入出力）
H－８８１１　Ｇｒａｐｈｉｃ Ｐｒｏｇｒａｍｍｉｎｇ Ｒｏｕｔｉｎｅ （GPR）
その他　（ローダー等）

4.3　プログラムの概要

このプログラムは，点と直線とによって表わされる三次元図形（wire frame）をCRT上に表示し，その図形に対し次のような操作を行うことができるものである。

(1) ユーザの指示に従って座標を設定し，その座標に合わせて入力された図形データを表示する。

(2) 図形を X, Y, Z 軸を中心として，ダイヤル等を用いて任意に回転させる。

(3) 適当な視点からの透視図を表示する（Z 軸上）

(4) 図形の拡大，縮少を行う。

(5) 図形の移動を行う。

(6) 図形の正投影，斜投影を行う。

これらの図形操作を行うには後章に述べてある同次座標を用いるのが便利である。

三次元座標のある一点〔 x, y, z 〕にもう一つの要素を加え，〔$x, y, z, 1$〕として表わしておけば，この点に対する平行移動，拡大・縮小，回転，透視図，正投影，斜投影，点対称等は次のような4行4列のMATRIX を乗じてやるだけで可能になる。更に便利なことは，これらのMATRIX を次々と乗じて行けば図形を回転して透視図を得る等の多重操作が非常に容易にできることである。

(1) 平行移動

$$\begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ e & f & g & 1 \end{bmatrix} = \begin{bmatrix} x+e & y+f & z+g & 1 \end{bmatrix}$$

(2) 拡大，縮小（0点を中止とする一様なスケーリング）

$$\begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & S \end{bmatrix} = \begin{bmatrix} x & y & z & S \end{bmatrix} \sim \begin{bmatrix} x/S & y/S & z/S & 1 \end{bmatrix}$$

$S>1$：縮小

$S<1$：拡大

(3) 回転

X軸のまわりに正の向きに θ 回転

$$\begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} x' & y' & z' & 1 \end{bmatrix}$$

(4) 透視図

視点はZ軸上hの距離

$$\begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1/h \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} x & y & 0 & 1+\frac{z}{h} \end{bmatrix} \sim \begin{bmatrix} \frac{x}{1+\frac{z}{h}} & \frac{y}{1+\frac{z}{h}} & 0 & 1 \end{bmatrix}$$

(5) 三面図（立面図）

$$\begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} x & y & 0 & 1 \end{bmatrix}$$

(6) 斜投影（X軸に対して）

$$\begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} a & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} ax & y & z & 1 \end{bmatrix}$$

(7) 原点対称

$$\begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} = \begin{bmatrix} x & y & z & 1 \end{bmatrix} \sim \begin{bmatrix} -x & -y & -z & 1 \end{bmatrix}$$

このプログラムは次の5つのファイルを持つ。

F0　図形表示命令用（線分の命令等）

F1　原図形データの保存用

F2　操作途中の図形データ

F3　変換され，現在表示されている図形データ

F4　CRT座標（ラスター値）に変換した図形データ

ここでF2が必要な理由は，ある図形に対して多重の操作を連続的に行いたいことがあるからである。またF1はもう一度元の図形を眺めたい時に必要になる。

これらのファイルのうちF0は他の4つのファイルの形式と違い「2点を結ぶ線分」の情報がストアされている。

他の4つのファイルはすべて，点を表わす三次元座標〔x，y，z〕の配列である。

これらのファイルをもとにして，前述したような図形操作を行うには，次のようなルーチンが必要になる。

(1) 制御ルーチン

(2) ファイル移動ルーチン

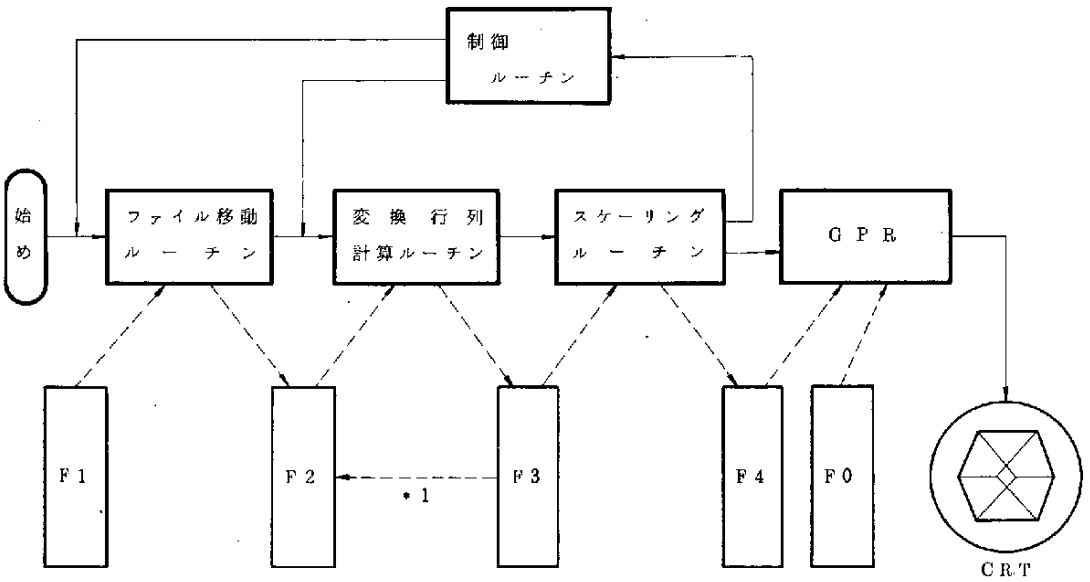
(3) 変換行列計算ルーチン

(4) スケーリング・ルーチン

(5) 座標軸表示ルーチン

そのほかに，実際に画面上に表示するコマンドをジェネレートしてディスプレイしたり，表示されている画面をX-Yプロッタにハード・コピーするためのルーチンとして前記のＧＰＲがある。

これらのルーチン間の関係を概念的に示すと下図のようになる。



Ｆ０　図形表示命令用ファイル

Ｆ１　原図形データ保存ファイル

Ｆ２　連続操作用中間ファイル

Ｆ３　変換結果ファイル

Ｆ４　ＣＲＴラスター値にスケールされたファイル

＊１　連続処理の際，変換，被変換ファイルを切換える。

ここで制御ルーチンの役割についてさらに説明すると，このルーチンは次のような機能を果すルーチンで，プログラム全体の流れをコントロールする。

(1) ファンクッション・キー，ダイアル，ライト・ペン，コンソール・タイプライタからの割込を受けつける。

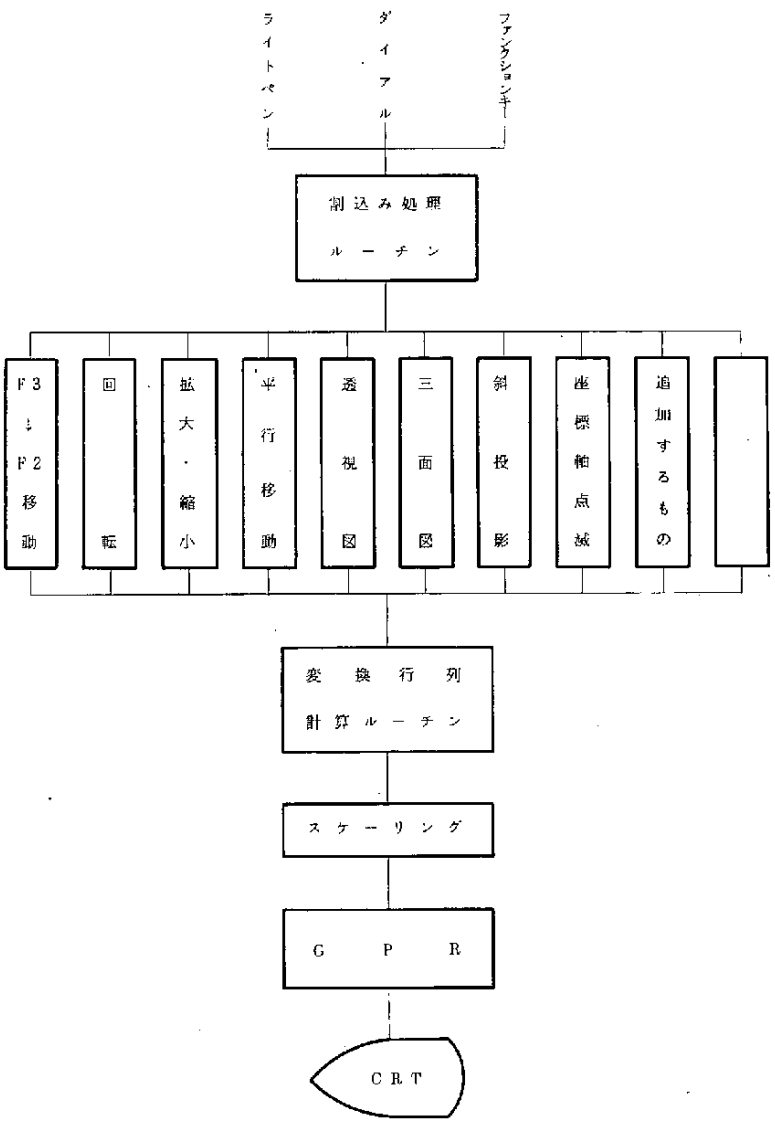
(2) それらの要因を解析し，夫々の処理ルーチンに必要なデータを渡す。

(3) 各処理ルーチンは，例えば回転の場合ダイアルの角度を読み取り，その値を変換行列の適当な要素にセットする。

(4) 変換ルーチンあるいは初期状態にコントロールを移す。

各処理ルーチンは，夫々ファンクッション・キーに対応させてあり，各ルーチンが制御ルーチンにより呼ばれる。また新しいファンクッションは必要に応じて追加できるようなオープン・システムになっている。

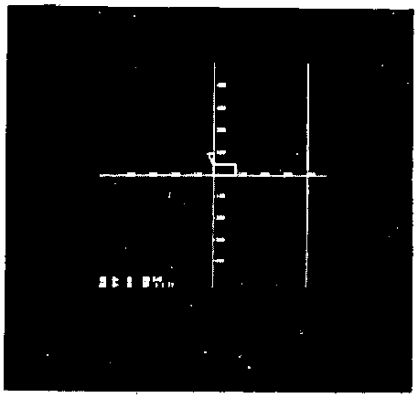
プログラム全体の構成は次のようになっている。



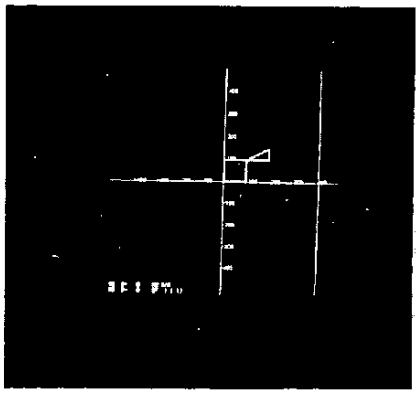
4.4 実 験 例

以下の例は本プログラムを用いてＣＲＴに表示した各種の図形である。

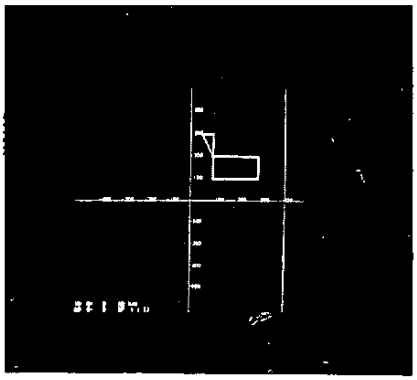
(1) 原図形の立面図



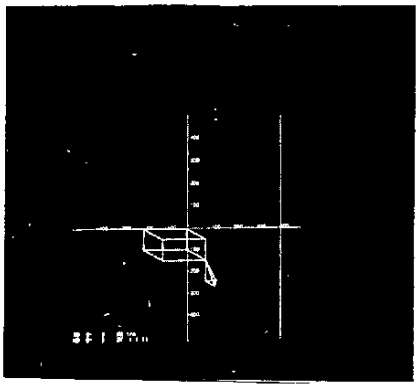
(2) ２倍に拡大した側面図



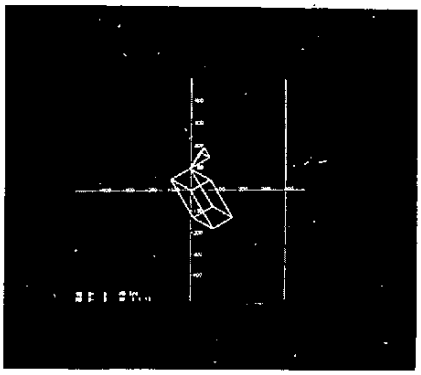
(3) Ｘ軸 Ｙ軸方向に平行移動



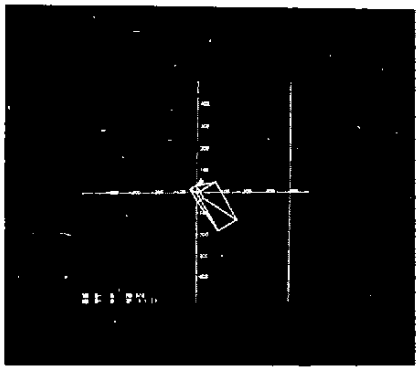
(4) 原図形の２倍拡大図をＸ軸,３０度,
Ｙ軸６０度回転



(5) (4)をさらにＺ軸６０度回転



(6) (5)の透視変換（視点Ｚ軸の手前）



4.5 問 題 点

現時点では実験の経験がまだ浅く，3．1で記した目標が完全に達成されたとはいえない。

しかし，これだけのプログラムをつくるのにも数多い苦労があったことは事実である。

これら実験的プログラムの作成の経験によって，得られた問題点（特にインタラクティブ・グラフィックスという観点から）は次のようなものである。

(1) 紙テープ・ベースのシステムであるということ

各システム・プログラム，処理プログラム，ユーザ・プログラムのローディレグを紙テープから行い，システムのイニシャライズ時に手操作による命令セットを必要とする等ということは，インタラクティブ・システムということから遠く離れており，非常に使いにくく，わずらわしい。このことは致命的な問題点である。

(2) プログラミングについて

これは(1)の問題と関連することだから，数百ステップ程度のソース・プログラムを紙テープにパンチし，ミス・パンチを修正し，コンピュータに読込み，ソース・リストがコンソール・タイプライタに印刷され，オブジェクト・プログラムが紙テープに出力されるのに，アセンブルだけで数時間を要するということ――これはプログラミングの効率ということから大きな問題である。

また浮動小数形式の演算命令がないということは，ちょっとした計算にも，非常にやっかいな処理を要する。それに，ページングということを，ユーザがプログラムを組む際に意識しなければならないということなども大きな問題点である。

(3) システム・キャパシティについて

記憶容量16KWということはやはり大きな制限である。

データ・ベースを持つことができないということから，大量のデータを必要とするアプリケーションには殆んど使用不可能となる。

これらの問題点は当初からある程度予想はされていたことであるが，インタラクティブ・グラフィックスということを実現するには，どうしても，もっと大規模なホスト・コンピュータ・システムに結合して，システム・キャパシティの問題を解決しなければならない。それに，H－8811グラフィック・システム側が紙テープ・ベースであることによるオペレーションの繁雑さは，ディスク・バック等を附加してシステム・プログラムをディスクからローディングするようにするか，ホスト・コンピュータ側からサテライトのH－8811側にイニシャライズ時に転送する等により解決しなければならない。

また，マルチ・コンピュータ・システムにしたとしても，H－8811側でのシステム・プログラムの作成ということは大きな負担となることなので，もっと便利なコンピュータでH－8811のアセンブラを作成し，そのプログラミング効率をあげなければならない。

5．X－Yプロッタの基本パッケージ

この章では，我々が開発したX－Yプロッタの Basic Soft について概要と簡単なアプリケーション（プロッタのSoftではむしろBasic Soft と言えるかもしれない）プログラムについて述べてある。Basic Soft については特に基本的な問題である，①直線を引くこと ②文字を描くこと，の2点について簡単な説明を加えておいた。

アプリケーションプログラムとしては破線を描くプログラムについて，フィッティングの問題を中心に述べてある。

5.1　ハードウェアに関して
（その概要）

現在X－Yプロッタにはディジィタル型，アナログ型，ハイブリッド型など色々あるが，我々が使用したのはディジィタルプロッタである。ディジィタル型のプロッタとは，Penがincri-mentalに移動しながら図形を描いていく方式のもので，1stepで動くPenの方向と長さが決っていて，方向は8方向，16方向，32方向などがある。1つのStepは0.1㎜とか0.25㎜と一見細かい動きをするが，実際に図を見るとギザギザした感じは免れない。（ハイブリット型式のものではPenの動きがまったく異っており，非常になめらかな線が描ける。）

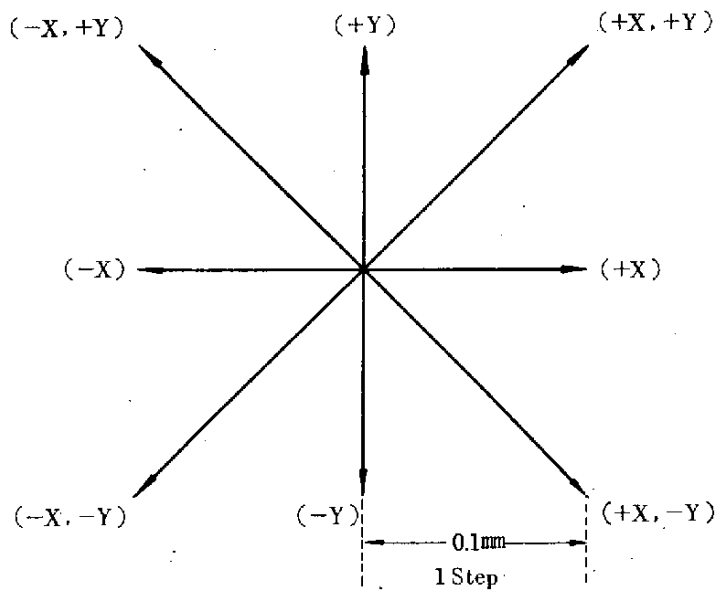


図1－1　ペンの動き（インクリメンタル方式）

インクリメンタル方式（ディジィタル方式）のものはX方向，Y方向の両方にPenが移動する。いわゆるフラットベット式とY方向のみペンが移動しX方向の移動はプロットシート自身が動いてこれに変えているドラム式がある。一般に小型の物にドラム式が多く，大型の物にフラットベット式が多い様である。

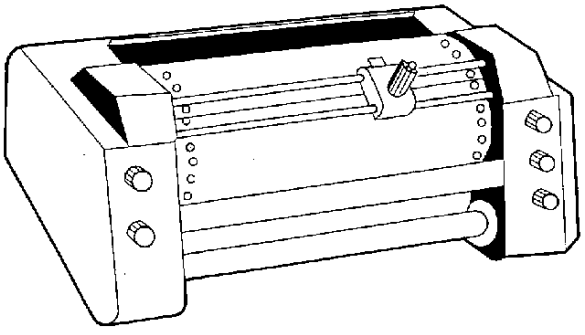


図1－2　ドラム式プロッタ

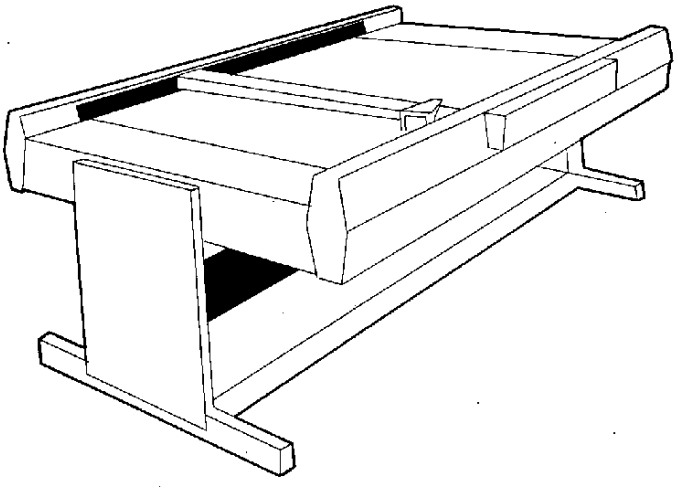


図1－3　フラットベット式プロッタ

どちらの場合もPen を移動させたり，ドラムを回転させる為にパルスモータが連動しており，入ってきたパルスの数に応じてPen やドラムが変位する様になっている。

（他の方式ではサーボモータなどを使ってPen を動かしている。）

さてプロッタをcomputerのoutput deviceとして実際に動かすにはon－line 形式とoff－line形式が考えられるが，X－Yプロッタの速度が遅い為にon－line で動かすには，multi処理でもしない限り非常に能率が悪い。そこで普通はプロッタに専用の磁気テープ・リーダが付いていて計算機で書いたテープを読みながらoff－lineでプロッタを駆動している。磁気テープ上の各レコードには実際にプロットする為のデータ（各方向を選ぶcommand data）の他にプロッタと同期を取る為の特別なコードが付いている。

（カルコンプの場合Sync Code, Pre sync code と呼んでいる。）

この様なレコードを幾つか集めて先頭に，Block addressと言ってこれらのレコード群を区別する為の番号を書いたBlockを置き，後で一部分だけ選んで図をプリント出来る様にしてある。

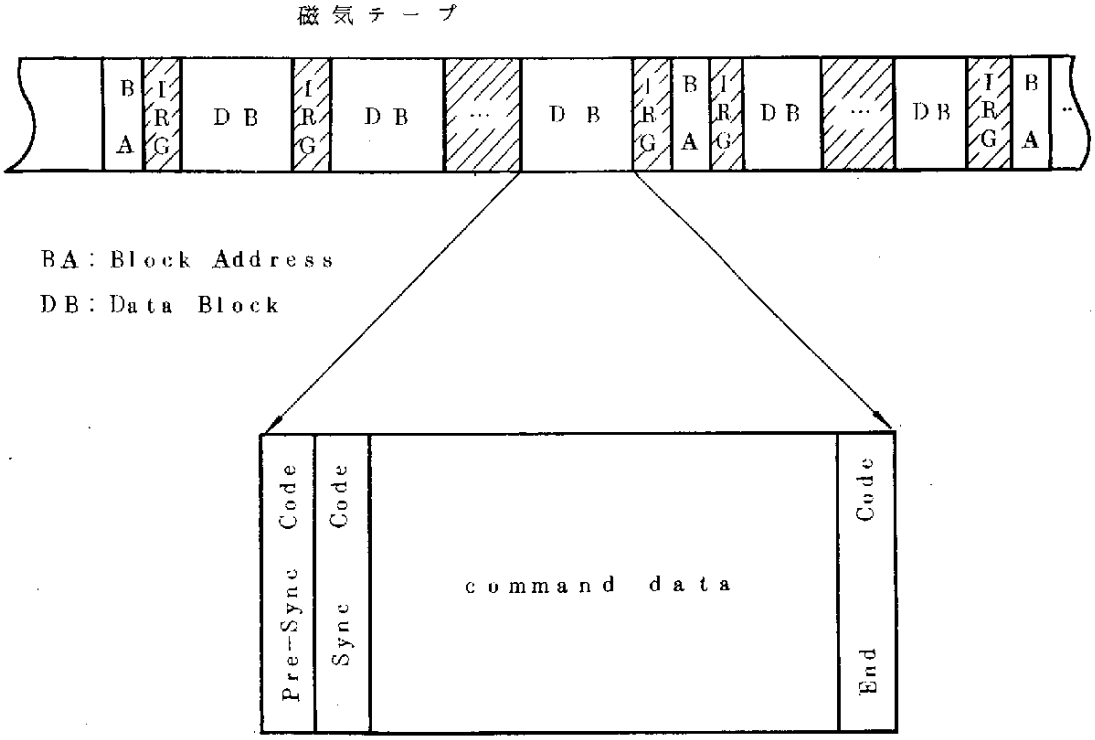


図1－4　Output Tape Format

プロッタをon-lineで駆動する場合データは純粋に(＋X，＋Y)，(－X，＋Y)，(＋Z　penup)など実際にプロッタを動かす様なcommandのみから成っている。ちなみに例をあげると(＋X，＋Y)などの，1stepのペンの動きに対応するデータは6bitで構成されており次の様である。

データ（8進表現）	プロッタの動作
02	＋X
03	－X
10	＋Y
14	－Y
12	＋X，＋Y
16	＋X，－Y
13	－X，＋Y
17	－X，－Y
20	－Z（pen up）
40	＋Z（pen down）

これらのcommand data が適当な大きさにBlockされて，例えばマルチプレクサーチャネルを経由してプロッタ（実際には入出力結合装置）に転送されそこで1つ1つのcommand data を解読しながら順にpen を移動させていく。

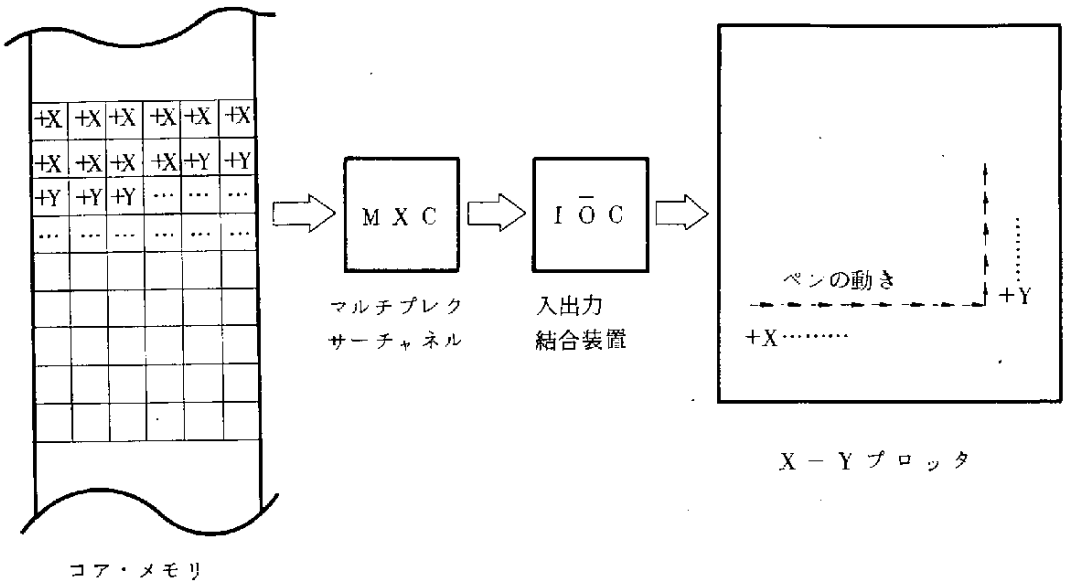


図1－5　プロッタをOn-line で駆動する場合の関係

5.2 ソフトウェアに関して

（その概要）

5.2.1 X－Yプロッタ用サポートルーチン

X－Yプロッタ用のサポートルーチンはFORTRAN プログラムの中からCALL形式で呼べるサブルーチン群より成っている。

user はコントロールカードの指定により

1. OFF-LINE 駆動用の磁気テープを作成する
2. 直接プロッタをON-LINE で駆動する

ことが出来る。

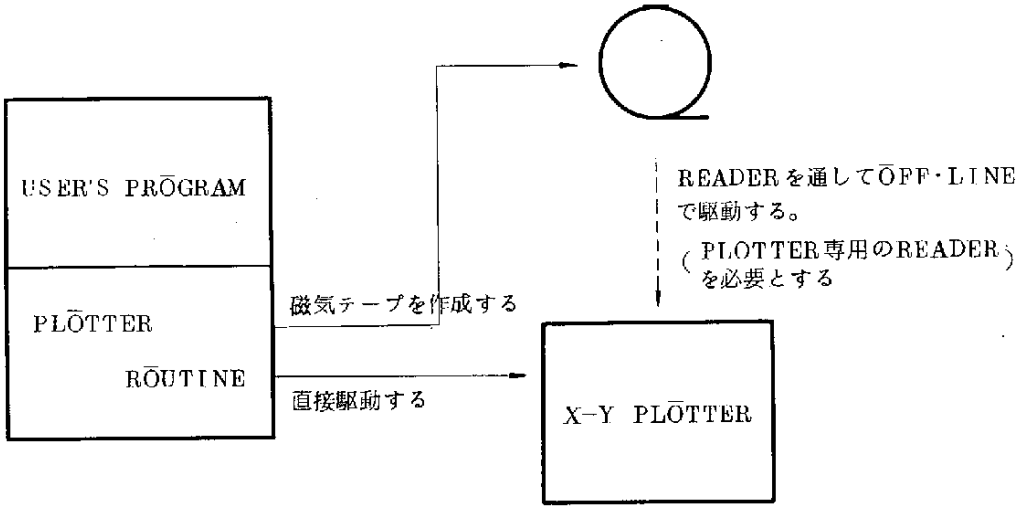


図2－1

サブルーチン群には

1. 任意の方向にペンを直線移動させる。……（PLOT）
2. 英数字特に文字を描く。……（SYMBOL）
3. 与えられた点列を直線で結ぶ。……（LINE）
4. データのスケーリングを行う。……（SCALE）
5. 座標軸を描く。……（AXIS）
6. 図形の倍率を変更する。……（FACTOR）
7. ペンの現在位置と倍率定数を知る。……（WHERE）

8. 原点の移動を行う。……(PLOT)

9. 与えられた浮動小数をPLOT する。……(NUMBER)

と言った機能を持つルーチンが含まれており，次に各ルーチンのCALLING SEQUENCE とパラメータの意味を簡単に記しておく。

1. PLOTS

CALL PLOTS(BUFFER(1) IBUF)

▲X－Yプロッタルーチンに対するOPENをする。

▲BUFFER(1): OUTPUT BUFFERの先頭番地

▲IBUF: BUFFER SIZE

2. PLOT

CALL PLOT(X, Y, IPEN)

▲現在ペンのある位置から点(X, Y)にペンを直線移動する。

▲IPEN: 移動の際ペンをUP 或はDOWN で行う為のコントロール用。

3. SYMBOL

CALL SYMBOL(X, Y, H, NBCD, TH, N)

▲NBCDで指定された文字列(N文字)を高さHで点(X, Y)よりX軸とTHの角度でかく。

4. LINE

CALL LINE(XX, YY, N, K, J, L)

▲配列XX, YYで与えられたN個の点を直線で結ぶ。指定があれば与えられた点にシンボルをプロットする。

▲K：XX, YYからデータを取り出す間隔

J：データ点の結び方の指定

L：データ点に描くセンタ・シンボルの番号

5. SCALE

CALL SCALE(XX, W, NN, K)

▲配列XXに入れられたNN個のデータをグラフに描く為スケーリングを行う。

▲W：データを収めるべき軸の長さ

K：XXからデータを取り出す間隔

6. AXIS

CALL AXIS(X, Y, NBCD, N, W, TH, XMIN, DX)

▲グラフの座標軸を描く。軸には名称，目盛，目盛ごとの数値を付ける。

▲NBCD: 軸の名称

N：NBCD のキャラクタ数

W：軸の長さ

TH: 軸の傾き

XMIN：軸の始点に対応する数値

DX： 軸の1インチに対応する目盛値

7. FACTOR

CALL FACTOR(FCT)

▲図形全体の拡大，縮小の尺度因子を与える。

8. WHERE

CALL WHERE(X, Y, FCT)

▲X，Yにペンの現在位置を，FCT に尺度因子を入れてくれる。

9. NUMBER

CALL NUMBER(X, Y, H, FPN, TH, M)

▲FPN で指定された浮動小数をプロットする。

▲X，Y：数値をプロットする位値

H：プロットする文字の高さ

TH： プロットする文字のX軸となす角度

M：プロットする数値の小数点以下の桁数

以上のルーチンの他に座標変換を行うOFFSET ルーチンなどがある。尚プロッタルーチンcloseはPLOT ルーチンをIPEN=999でcall する事により行う。

これらBasic SubroutineはSYMBOL, LINE, AXIS など実際図形を描くSubroutineは全てPLOT ルーチンを使ってこれを行い，自分自身でPLOTTERのCommand を作り出す事はない。

Basic SubroutineはPLOT ルーチン，つまり直線を描くルーチンを基本に組立てられており，I/O の管理，座標系の管理など一切，このルーチンで行っている。

各ルーチンの関係を図に示すと次の様に成る。ここに矢印の意味は，A→ BならAルーチンの中でBルーチンを作っている事を意味する。

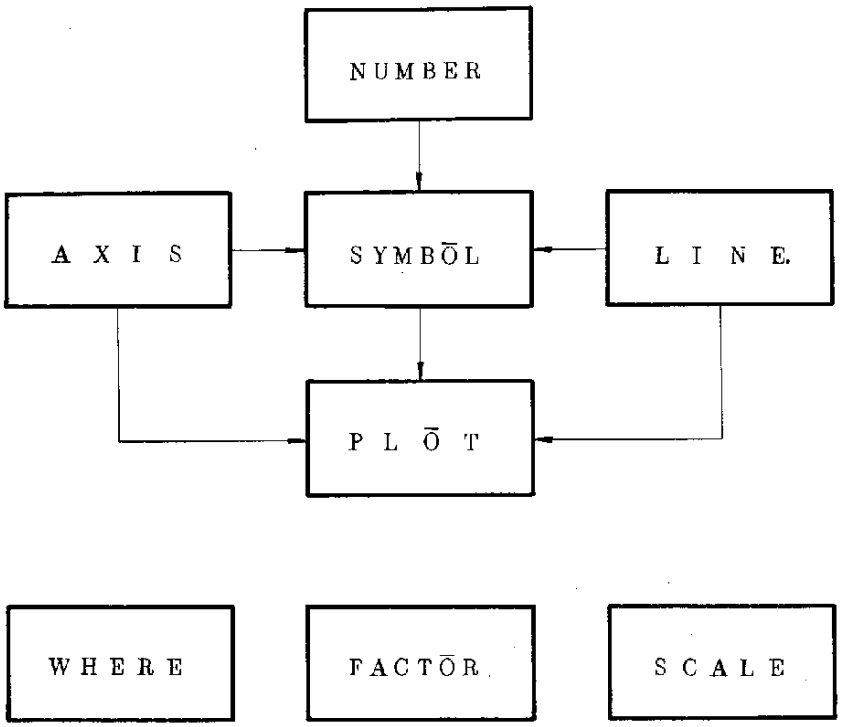


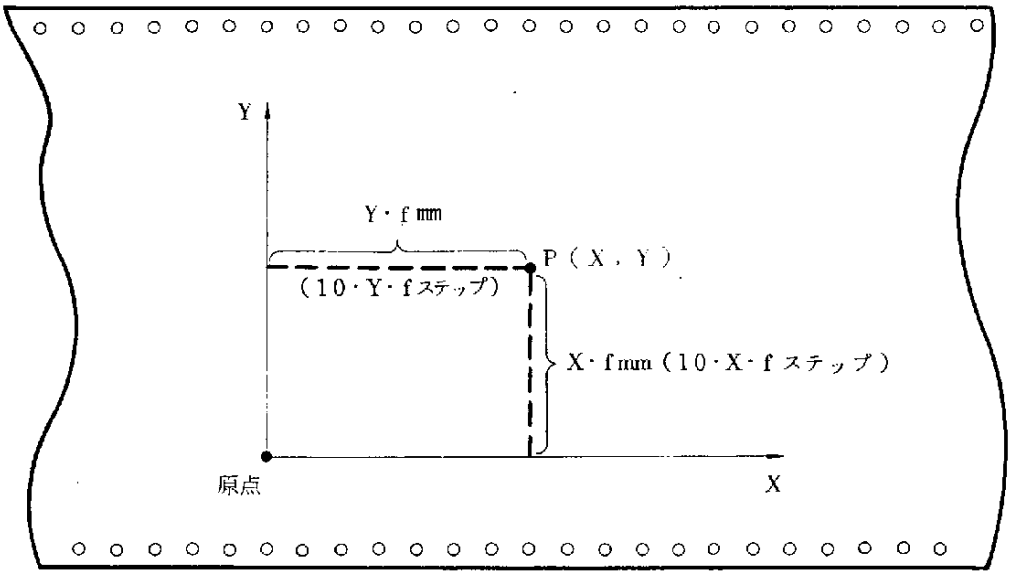
図２－２　サブルーチンの関係

5.2.2　サポートルーチンに関する基本問題

A. 座標とペンの位置

userがサポートルーチンを使う時に指定する点P（X, Y）とプロットシート上にあるペン位置との関係は次の様である。

1. userがFACTORルーチンによって尺度因子fを指定していなければ，原点よりX軸方向にXmm, Y軸方向にYmmの位置に対応する。
2. 尺度因子fを指定してある場合には原点よりX軸方向に$X \cdot f$mm, Y軸方向に$Y \cdot f$mmに対応する。



プロット　シート

図2－3

我々の使ったプロッタの1Step－0.1mmなので，点P（X，Y）は原点よりX軸方向に

〔10・X・f＋0.5〕ステップ，Y軸方向に

〔10・Y・f＋0.5〕ステップの位置に対応する。

（〔　〕はガウス記号）

この様に原点よりのステップ数で測る座標を以後absolute座標と言う事にする。

B．ペン位置の管理

プロッタはハード的にPen の位置を知る機能を持っていない。（但しSide end，Paper endの検出は出来る）そこでPen の位置は全てSoft で管理しなくてはならない。

1. 原点

Pen が始めにあった所，正確には，PLOTSルーチンがCALLされた時点で，Pen のある位置を原点とする。

2. その後のPen の位置

プロッタ管理ルーチンの中で実際にPen の移動を行うのはPLOTルーチンのみである。（他の場合，例えばSYMBOLルーチンなども実際に文字を描くときには，PLOTルーチンをCALLする。）

そこでPLOTルーチンでX方向，Y方向にコマンドの数を憶えておけば，現在Pen がどこにあるかを正確に知る事が出来る。

C．ペンの直線移動

前にも述べた様にX－Yプロッタは与えられた8方向のコマンド列に従ってペンを順次移動させていく。仮にペンをX軸方向，或はY軸方向に直線に移動するには（＋X）のコマンド，或は（＋Y）のコマンドをつらねて置けばよい。しかし任意の方向に直線移動を行うには8方向のコマンドを適当な順に繰り返して，最もその直線に近い様な列を作らなくてはならない。つまり8方向以外の直線は実際直線ではなく折線によって近似を行っている。

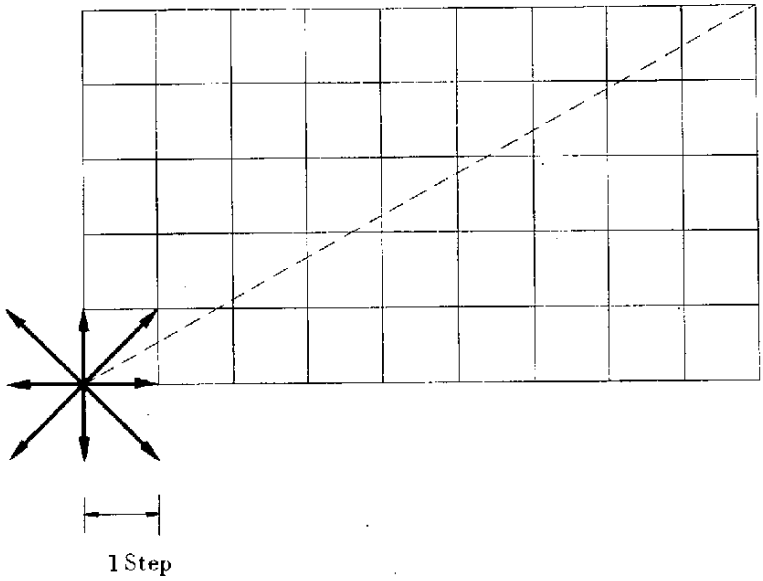


図2－4

D．直線近似の方法

直線近似は次のルールにより原則として，2種類のコマンドの繰り返しで行っている。

ある始点から,指定された終点まで直線を引くには，始点を中心にコマンドと同じ8方向の向きを持つ直線を引く。各直線にはもちろん対応するプロッタのコマンドがある。そこで終点をはさむ2直線に対応するコマンドの組み合わせで，この線分を折線近似する。終点が直線上にあれば，明らかにそれと対応するコマンドだけで，直線がかける。

今のabsolute座標に於ける点P_A（IXA，IYA）から点P_B（IXB，IYB）まで折線近似することを考える。ここで

$$IDX = IXB - IXA$$

$$IDY = IYB - IYA$$

とし，$IDX > 0$，$IDY > 0$かつ$IDX \geqq IDY$と仮定する。すなわち点P_AからP_Bまでの接線はX軸方向のコマンド（＋X）と45°方向のコマンド（＋X，＋Y）で構成する。

コマンドは次のルールによって順次選択する。

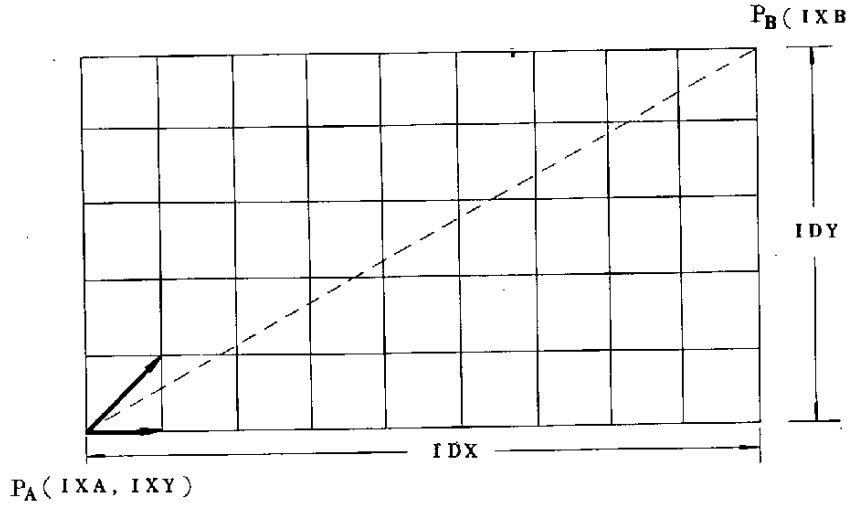


図２－５

① 点P_Aに於て直線P_AP_B と$X=IXA+1$を通る軸との交点が$IYA+1/2$ 以上なら（＋X，＋Y）を，他の場合には（＋X）を選ぶ。（IDX≧IDYを仮定しているので交点がIYA＋１を起える事はない。）

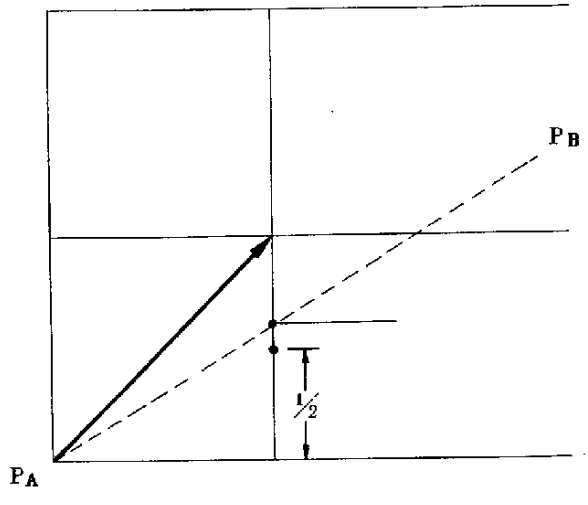


図２－６

② 直線P_AP_Bと$X=IX+n_x$（n_x=2, 3, ……）軸との交点に於いて順次，次のような方法でコマンドを選ぶ。

点P（$IX+n_x$, $IY+n_y$）まで折線で近似した時，直線P_AP_B の$X=IX+n_x+1$ を通る軸との交点のY座標をL_y とする時

$L_y-(IY+n_y)\geqq 1/2$

ならばコマンド（＋X，＋Y）を，そうでなければコマンド（＋X）を選ぶ。

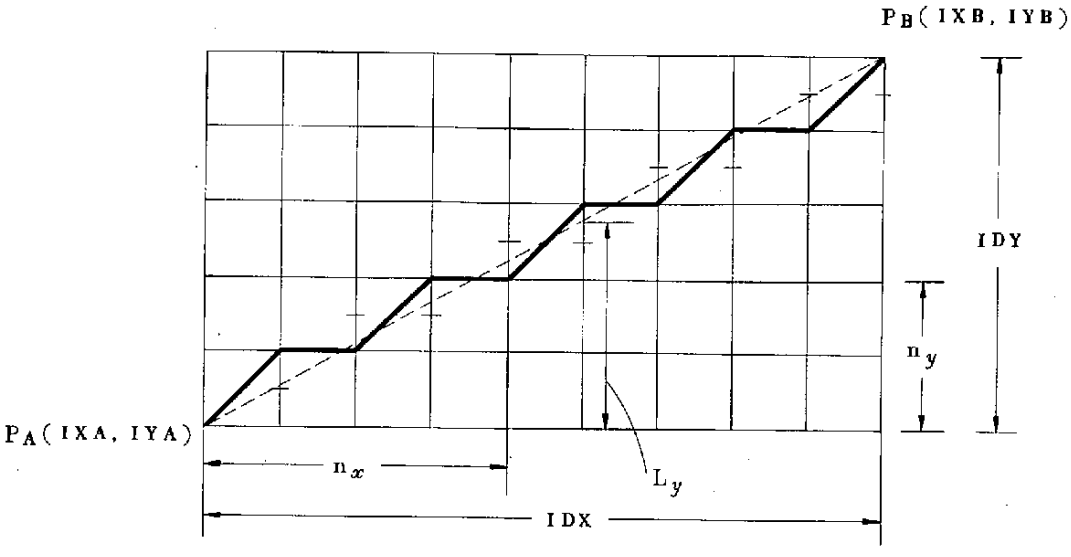


図２－７

E．PLOTルーチンでの処理

現在Pen の在る位置（始点）から，パラメータで指定された終点（X，Y）まで直線動を行う為，プロットルーチンは次の順で処理を行っている。

1. 終点（X，Y）をabsolute座標に変換する。
   （IX，IYとする。）
2. Pen の現在位置をabsolute座標で表したものをIPX，IPY とすると，その差
   IDX＝IX－IPX
   IDY＝IY－IPY
   を計算する。
3. IDX，IDY の値により2方向のコマンドを選択する。
   座標軸方向のコマンドをCA，他方をCBとする。
4. |IDX|，|IDY| のうち大きい方をNC，他方をNRとするとコマンド列は次のFLOW CHARTの様にして決定される。

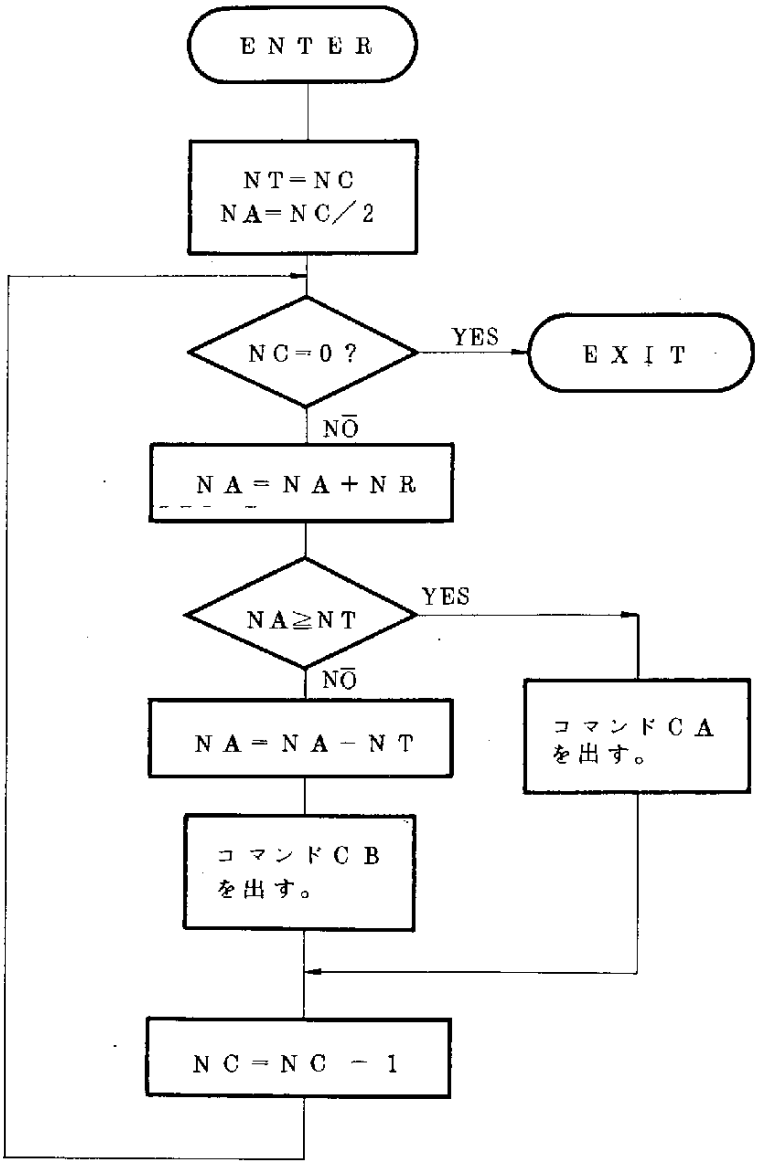


図2－8　折線近似のFLOW CHART

F．文字を描く事

最近の大きなX－Yプロッタでは，Pen と一諸にSymbol Printer を持っていて，指定された文字を直接プリント出来る様になっているものがある。しかし一般のプロッタではこの様なSymbol printer は付いておらず，又これだけでは文字の大きさが何時も同じで自由に角度を変えてプリントする事も出来ない。そこで普通のPen を使って文字を書く機能がどうしても必要になってくる。

我々のシステムではSYMBOL というルーチンにより指定された文字列を位置，大きさ，角度などをサブルーチンのパラメータで指定してプロット出来る様にしてある。文字の形体はそれぞれ決っていて，次の表はSYMBOL ルーチンでプロットする文字とそれに対応するCode（EBCDICコードのoctal表現）をプロッタで描かせたものである。

TABLE 2-A

CHARACTERS AVAILABLE IN SYMBOL ROUTINE〔FACOM 230/60〕

BI-OCTAL EQUIVALENCE
REFERENCE LOWER LEFT CORNER

301	A	321	J	342	S	360	0
302	B	322	K	343	T	361	1
303	C	323	L	344	U	362	2
304	D	324	M	345	V	363	3
305	E	325	N	346	W	364	4
306	F	326	O	347	X	365	5
307	G	327	P	350	Y	366	6
310	H	330	Q	351	Z	367	7
311	I	331	R	140	‾	370	8
100		132	!	141	/	371	9
113	.	133	$	153	,	172	:
115	(	134	*	157	?	175	'
116	+	135	)			176	=

INTEGER EQUIVALENCE
REFERENCE CENTER

0	□	16	!
1	○	17	↓
2	△	18	↑
3	+	19	≡
4	×	20	→
5	◇	21	%
6	☆	22	≥
7	⧖	23	^
8	Z	24	≠
9	Y	25	±
10	⌑	26	0
11	✳	27	1
12	⧖	28	2
13	|	29	3
14	☆	30	4
15	—	31	5

図２－９

1. 普通文字は４×７の格子を考えて左下隅を原点とした格子点の座標で表わす。

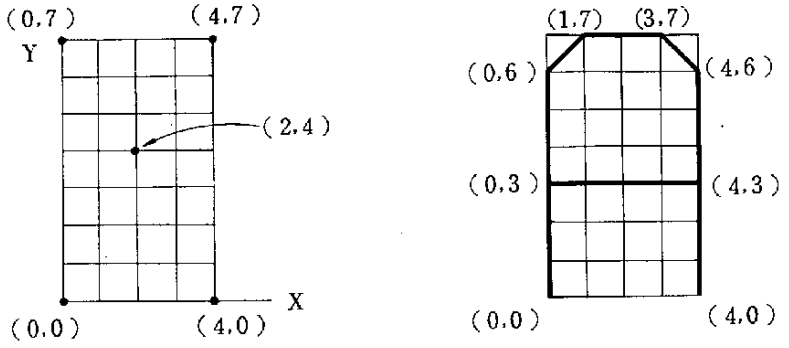


図２－１０　格子座標と文字の構成

例えば英字Ａは図の様に表わし，結んでいく順に格子点の座標を並べておく，つまり

（ 0,0 ）（ 0,6 ）（ 1,7 ）（ 3,7 ）（ 4,6 ）（ 4,3 ）（ 0,3 ）（ 4,3 ）（ 4,0 ）

と言う様にする。

2. 文字の高さ，幅，間隔（幾つか文字をかく場合）はいつも一定の比率にしてしまう。高さHが与えられると幅は（ ４／７ ）・H，間隔は（ ２／７ ）・Hとして計算する。

3. 決められた位置（ X_0 ， Y_0 ）からX軸とθの角度で高さHの文字を書くこと。

文字を構成する各点の格子座標を（ $X_{\overline{OF}}$， $Y_{\overline{OF}}$ ）としuser側の座標から見たこの点の座標を（ X， Y ）とすると次の関係が成り立つ。

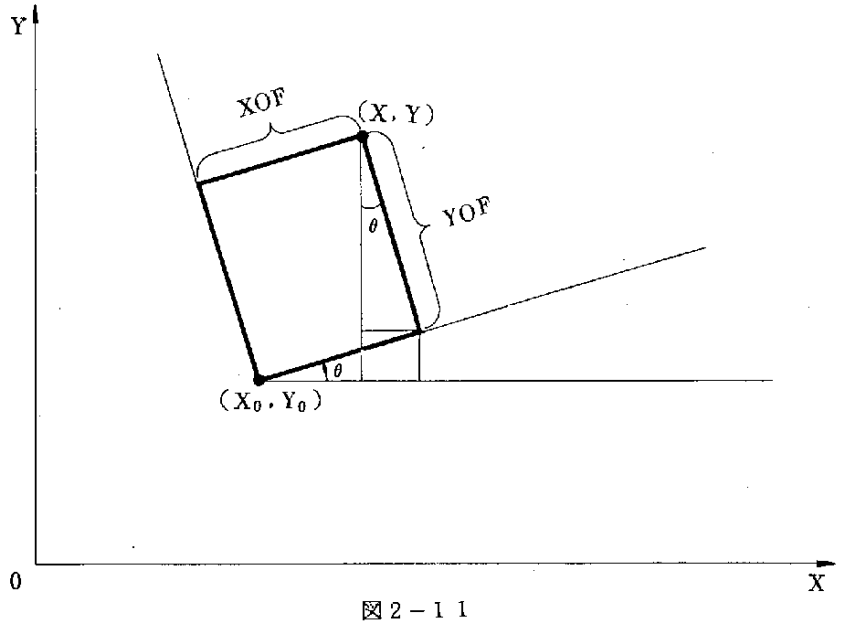


図２－１１

$$X = X_0 \ (\ X_{OF} \cdot \cos\theta - Y_{OF} \cdot \sin\theta \ ) \cdot H$$

$$Y = Y_0 \ (\ X_{\overline{O}F} \cdot \sin\theta + Y_{\overline{O}F} \cdot \cos\theta \ ) \cdot H$$

そこで与えられた格子点の座標から順次，X，Yを計算して直線で結べば文字が書ける。

2つ以上の文字ｓｔｒｉｎｇ を描く時には

$$X_0 = X_0 + \frac{6}{7} H \cdot \cos\theta$$

$$Y_0 = Y_0 + \frac{6}{7} H \cdot \sin\theta$$

とX_0，Y_0 を修正して同じ方法で次の文字がかける。以上同じ様に繰り返せばよい。

5.3 簡単なアプリケーションの一例

5.3.1 破線を描くプログラム

プロッタでグラフなどを描かせる時，線を区別する為に色々なSymbol を線上に描かせる事があるが，実線，破線，一点鎖線などによってこの区別をしたい事もある。

この様な機能を持ったプログラムについて色々調べて見ると特に破線，一点鎖線，二点鎖線などを別のプログラムで描かせなくてもほとんどどれも，同じ方法で描ける事が解る。

そこで我々のプログラムではuserに任意のパターンを指定させてどんな線でも自由に描けるプログラムを作った。

パターンについては，例えば破線なら実線部分の長さと空白の部分の長さを，この順で指定させ，一点鎖線であれば長い実線部分を始めに，次に空白，次に短い実線部分の長さを，最後に空白の部分を指定させる。

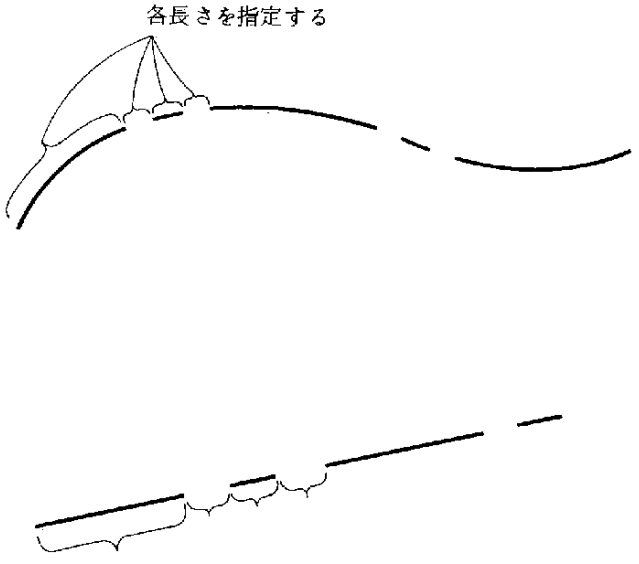


図3－1　パターン指定

この様に繰り返す部分のパターンを任意に指定する事が出来る。但し実際には指定した通りの長さにはならず，曲率が違うと長さも違ってしまう傾向がある。

この様なルーチンには大きく分けて2つの機能が必要である。つまり

1. 与えられた点列を適当な曲線でフィッティングする。
2. 適当なパターン（userが指定する。）に従ってPen のup, down の操作を行う。

5.3.2 Fitting

フィッティング（つまり与えられた点列P_1，P_2，P_3 ……P_n の各点をなめらかに曲線でつなぐ事）は「なめらか」と言うのがひどくあいまいな表現であって色々と問題の在る所である。我々のプログラム（破線を描く）では本来の目的が，破線により他の線とその種類を区別するのみで，特に精度を要しないものであった。その為隣り合うセグメント（2点間を結ぶ曲線をセグメントと言う）がその接続点に於てそれぞれの接線に等しい時「なめらか」であると決めている。

フィッティングに関する問題は色々発表されているが我々は次の様な方法で行った。

① 各点を通る直線（接線となるべき）を与える。

② 隣り合った2点に対して両点を通る3次曲線の接線（両点に於ける）が①で与えられた直線に成る様，係数を決める。

A．接線を与える事

接線を与える方法としては，現在次の2方法が良く知られている。

a．点P_i には，2点P_{i-1}，P_{i+1} を通る直線に平行な直線を与える。

P_1，P_n に対しては仮想の点P_0，P_{n+1} を考えて同様に扱う。

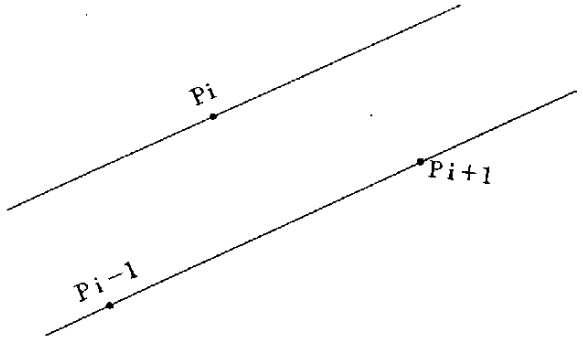


図3－2　直線の決め方

b．点P_i に与える直線は3点P_{i-1}，P_i，P_{i+1} を通る円を考えて，点P_i に於ける円の接線とする。

点P_1 に於てはP_1，P_2，P_3 を通る円のP_1 に於ける接線とし，同様に点P_nに於てもP_{n-2}，P_{n-1}，P_n の3点を通る円を考えP_n に於ける接線とする。

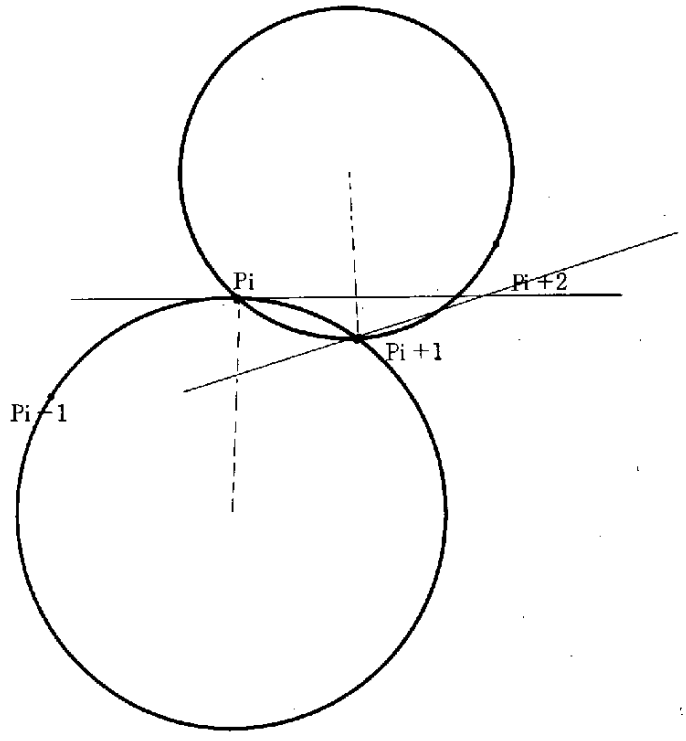


図3－3　直線の決め方

B．3次曲線の係数を決める事

各点に接ベクトルが与えられると，それをもとに3次曲線の係数を決める事になる。

破線,或は任意のパターンを描かせようと思うと，曲線上の長さを測り,それに従い，ペンのアップ,ダウン操作を行う必要がある。そこで出来れば曲線上の距離Sをパラメータとした3次曲線

$$P(S)=AS^3+BS^2+C\cdot S$$

（A，B，C，は定数ベクトル）

を仮定しその係数を決められると何かと都合が良い。

ところが両端に於ける接ベクトルの条件，及び通過する点の条件を入れて係数を決めようとすると全ての係数が0でなくては，この様な条件を満さないという矛盾が起ってしまう。（ベクトルの要素を実数とした場合）そこでSを近似的に曲線距離を表わすパラメータと考えて次の様な条件を考えてしまう。

今，どの2点を取っても条件は同じだから点P_1からP_2への3次曲線を決める事を考える。

① $S=0$の時P_1を通る。

② $S=S_0$の時P_2を通る。

ここでS_0は$\overline{P_1\ P_2}$で代用する。

③ $S=0$で$P'(S)$の方向がP_1に於ける接線と同じで，大きさは1である。

④　$S = S_0$ に於ても同じ様にP′(S)の方向は接線と等しく，大きさは１である。

これらの条件については①はともかく③，④の仮定はP(S)のSが正確に曲線P(S)の距臨を表わす時成り立つ式であり，③，④を正当に見ると②が近似的な表現となってしまう。又，逆にSは特に曲線P(S)の距離を示すパラメータでないと見れば②の条件も正しいが，③，④が正確に成り立つとは言えない。

もっと正確な条件により式を立てる事も出来るが，この様な仮定をすると処理が少なくてすむ利点があり，今回の目的から言うと充分である。又実際に曲線を描かせて見ても「なめらか」にフィッティングされている事がわかった。

ちなみに，条件からは次の様な式が成り立つ

1. 条件②より

   $P_2 - P_1 = A S_0^3 + B S_0^2 + C S_0$

2. 条件③より

   $P'(0) = C$

3. 条件④より

   $P'(S_0) = 3A S_0^2 + 2B S_0 + C$

ここで$S_0 = |P_2 - P_1|$で$P'(0)$，$P'(S_0)$はP_1，P_2に与えられた接線から傾きから計算出来，３つの式よりA，B，Cを求める事が出来る。

C．３次曲線を決定した後に破線を描く為にペンのup，downのcontrolを行う必要があるが，これは純粋にプログラミング上の問題だけなのでここでは省略する。

次にこのプログラムによりSIN curveを実線，破線，一点鎖線，二点鎖線で描いた例を付けておいた。

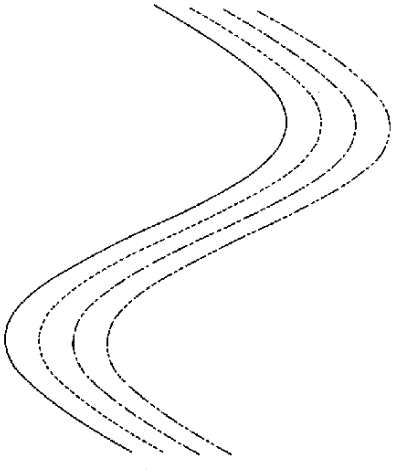


図３－４

あ　と　が　き

あ と が き

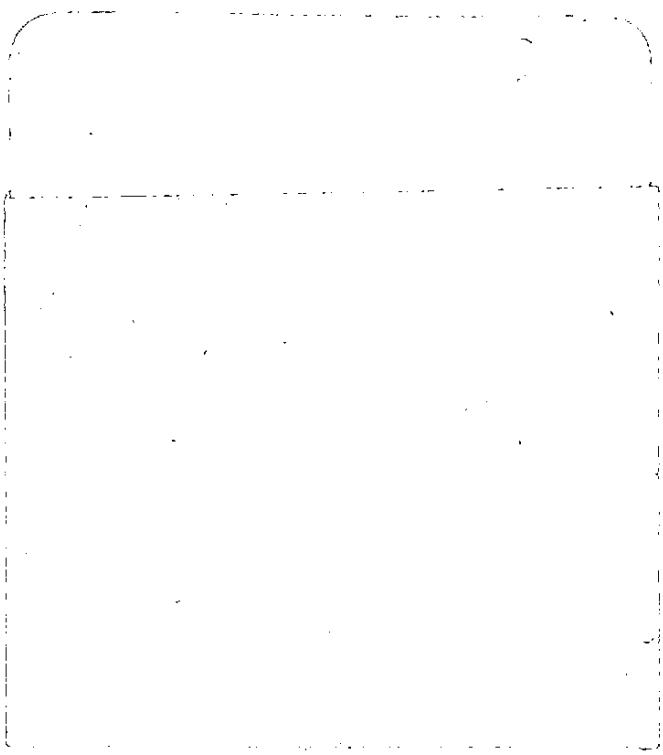
図形表示用のソフトウェアの実用化が進むにつれ，ハードウェア，ソフトウェアとも，さまざまな問題点が浮きぼりにされて来た。とくにグラフィック，ディスプレイは未だにその価格の高価な点も含めて，最初予期された程の実用化が進まず，また汎用グラフィック・ランゲージ，汎用ソフトウェアの開発もあまり進展を見ていない。

今後に残された問題は大きく且つ困難である。

禁無断転載

昭和45年3月発行

発行所　財団法人　日本情報処理開発センター
東京都港区芝公園21号地1番5
機械振興会館内
TEL（434）8211（代表）

印刷所　三協印刷株式会社
東京都渋谷区渋谷3－11－11
TEL（407）7316

44－S004