YN_COREのインプリメント

今宮淳美，坂本忠明，好澤一總，今田正卓
（山梨大学 工学部 計算機科学科）

1. はじめに

1.1 標準グラフィックス・パッケージ

グラフィックス・パッケージの標準化に関して、当研究会に於いても既に紹介され、関連した報告も種々あるので[1]ここでは簡単にグラフィックス・パッケージの標準化の流れと，その目的についてふれる。

グラフィックス・パッケージの標準化の動きは，1974年ACM/SigGraph-GSPCが設立されたことに始まり，1976年にSeillac(IA)に於いてコンピュータ・グラフィックスに関する基本的合意がなされ、1977年には，このSeillac-Spritを基にアメリカではGSPC-COREシステム、西ドイツを中心としたヨーロッパではGKSが提案された．そして1979年COREでラスター拡張を含む改訂版が提案された。又，同年からANSIに於いて5年計画でCOREとGKSの融合が計られている。

こういったグラフィックス・パッケージ標準化の動きはグラフィックス装置の低廉化やCAD/CAMを始めとするコンピュータ・グラフィックスの利用の増加等とあいまって『応用プログラムの可搬性』『プログラマの可搬性』を確保することでより広汎なグラフィックス利用を目指している．そしてその目的達成の為、以下の方法論が考えられた．

(1) 装置独立…グラフィックス装置の機能とは独立してグラフィックス機能を定義しておくことで応用プログラムの可搬性が確保される．

(2) グラフィックス・システムとモデリングの分離…応用プログラムに依存しないグラフィックス・システムの確立をはかる．

又．グラフィックス・パッケージ自体の可搬性を考慮するようANSIからVirtual Device Interface(VDI)の機能集合が提案されている．

1.2 研究の目的とアプローチ

以上のような状況をふまえて，ANSIのVDIを考慮した形で，YN_COREシステムを実現する．以下でその構成・特徴等を述べる．

システム実現には大きく3つのステップから成る．

(1) システム全体の仕様決定，
(2) ソフトウェア・デザイン，
(3) 実際的インプリメント．

(1)はACM_COREを基本にしており，COREレポート[2]の解析に多くの時間が割かれた．(2)及び(3)は後述される．

2. YN_COREの特徴

YN_COREシステムを作成する上で考慮した特徴点を列挙し各々に説明を加える．

(A) ACM/SigGraphが提案するCOREシステムの諸機能をサポートする．

但し，第1版では出力レベル3，入力レベル2，次元レベル2Dである．ラスター拡張は除く．従ってYN_COREは，出力ではイメージ変換属性や可視属性といったセグメント属性をもつ保持型セグメントの使用が可能であり，入力はsynchronousである．前述したようにCOREをサポートすることで応用プログラムの可搬性

を目指している.

(B) ANSI(X3H33)が提案するVDI functions[3]をDevice Independent/Device Dependent Interfaceとして用いる.

これにより、システム自体の可搬性も著しく向上するものと期待される. しかし現在提案中のANSI VDIはCOREのみに着目して作られたものでない為, いくつかの点でCOREになじまない機能が存在するので, YN-COREではいくつか変更している.

(C) Multi Devices をサポートする.

これは山梨大学計算機科に複数のグラフィックス装置があり, しかもその各々が特徴をもっている(例えば, D-SCANは高度なインテリジェント機能をもつが線分のカラー表示は7色であり、M303はRGB各8レベル512色表示可能である.)ために、上手くmulti devices を用いることで, よりよいman-machine interactionを期待できる.

(D) 各グラフィックス装置のインテリジェント機能を最大限活用する.

COREのESCAPE function を用いて, COREには定義されていない各グラフィックス装置固有機能の利用を可能にしておく.

(E) 既存のdevice driverを利用する.

既にdevice driverのインプリメントが終わっていたので, そのまま手を加えず利用することにした.

(F) システムの可搬性・拡張性を充分考慮した基本設計.

3. YN-COREシステムのソフトウェア構成

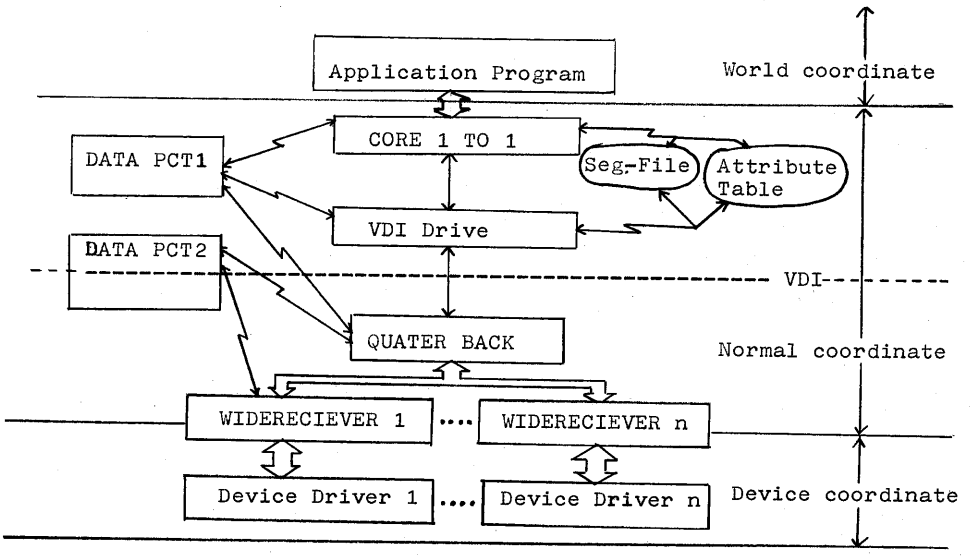

Fig.1 YN-COREシステムのソフトウェア構成

- 応用プログラム
  ---ユーザの作成するCORE機能列

- CORE 1 to 1
  ---主にattributesの変更とユーザの問い合わせに答える．DATAPCT1にCORE function codeと引数を置き、VDI Driveに操作を受け渡す．
- VDI Drive
  ---セグメント・ディレクトリを作り、unbuffered typeのグラフィックス装置の為セグメント内容を保持する．又．CORE functionを適時VDI functionに変換する．DATAPCT2にVDI function codeと引数を置き、Q.Bに操作を受け渡す．
- QUATER BACK （Q.B.）
  ---VDI functionをselect surface 情報に従ってディバイス・ソフトウェアへ出力する分配器の働きをする．
- WIDE RECIEVER
  ---DATAPCT2に置かれたVDI function codeを解釈し、各ディバイスに対応した機能に変換しディバイス・ドライバを呼ぶ．
- Device Driver （D.D.1,…,D.D.n）
  ---各々、D-SCAN，M303，MB501，T4010，XYプロッタを実際に駆動するルーチン群．

4. Virtual Device Interface[3)]

4.1 目的

CORE，GKS等はいわばtop-down的グラフィックス標準へのアプローチであり、VDIはbottomup的アプローチである．VDIを導入することにより、VDIより上位部分の可搬性を計る。

4.2 VDI Model

VDI functionsはRequired FunctionsとNon-Required Functionsに分類できる．

(1) Required Functions---D.D.がその機能をもっており，D.D.に問い合わせることなく利用可能なfunctionである．

(2) Non-Required Functions---D.D.がその機能を必ずしも持つ必要がなく，上位ブロックからD.D.へ問い合わせし，もしD.D.がNon-Supportedであった場合VDIの上でその機能をシミュレートする．

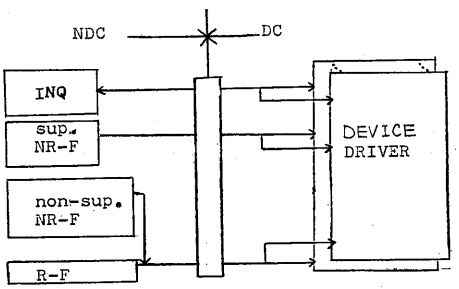

fig.2 VDI model

VDI func. / D.D. func.	required	non required
supported	callable	callable
non supported	emulate on D.D.	emulate on VDI drive

table 1 relation of VDI functions to D.D. functions

5. YN_CORE OUTPUT/CONTROL

5.1 TEXT

(1) TEXTルーチンの引数

引数は文字列であるが、文字列を1文字毎に分離・解釈するには、

ア) 文字列以外に文字数を表わす引数を導入する。(ex.) TEXT (n, 'string')

イ) 文字列の前後に特殊記号を付加する。 (ex.) TEXT ('*string*')

YN_COREでは、CORE提案書にある引数の形に変更を加えることを避ける為、イ)の方法を採用し、文字列を／でくくることにした。

(2) TEXTのセグメント保持

TEXT function code
character1
character2
⋮
charactern

fig. 3(a)

MOVE (x_0, y_0)
LINE (x_1, y_1)
⋮
LINE (x_n, y_n)

fig. 3(b)

ア) 図3(a)のような形でセグメントにTEXTコマンドを保持する場合、char-up等の変換情報も同時にセグメントに保持せねばならない。

イ) 図3(b)のようにTEXTコマンドをMOVE, DRAWのコマンド列に変換し、線図形としてセグメントに保持する場合、前述のようなchar-up等の情報は保持する必要がない。しかしもしVDIの上にセグメント・ファイルがあるとするとVDI機能にあるTEXT関係の機能は不要となる。又、データ量が多くなる欠点をもつ。

YN_COREでは、ソフトウェアでTEXTコマンドをサポートしている関係上、イ)の方法を採用している。

注 YN-COREで用意されている文字データは、図4のような33×41ドットの矩形中に於いて定義されている。

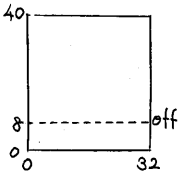

fig. 4

5.2 ビューイング変換

- ユーザはディバイス座標を意識することなく、W.C.上でモデルを構築・操作出来る。
- セグメント情報はN.D.C.で保持される。
- 一般的に2nd変換は図5のように行なわれる。ESCAPE functionによりセグメント内容表示をディバイス側でも変更出来る。

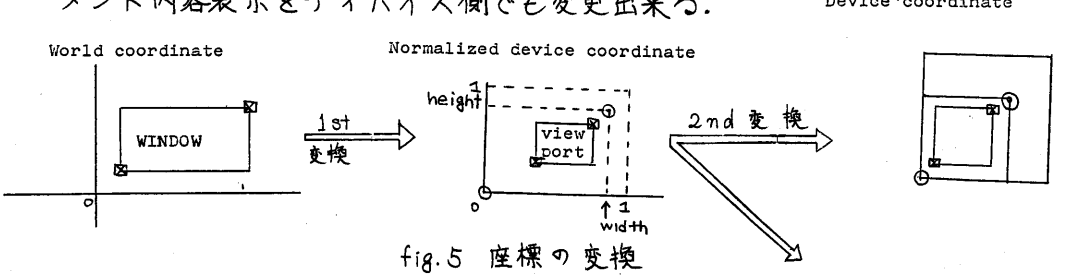

fig. 5 座標の変換

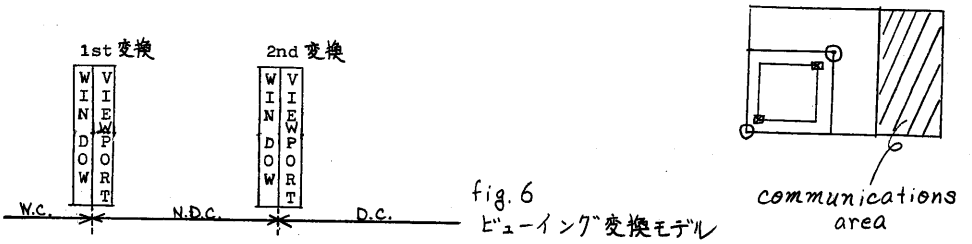

fig. 6
ビューイング変換モデル

5.3 セグメント

(1) VDIの上に設置するか．下に設置するか．

後述する．

(2) 主記憶上に配列としてとるか．ディスク上に外部ファイルとしてとるか．

データの操作性から試作段階では主記憶上に配列でとるが，実用段階では大量のデータが予期されるので外部ファイルを必要とすることが予想される．

(3) 構造

セグメントの属性等を保持するセグメント・ディレクトリ用配列と実際のfunction列を保持するセグメント内容用配列の2つがとられている．前者は各セグメント毎に属性を与えるのに容易なように2次元配列になっている．属性は表2のように保持される．

SEG-ARY1(1,I)----	Segment name	SEG-ARY1(7,I)----	Sy
(2,I)----	Surface name	(8,I)----	a
(3,I)----	Visibility	(9,I)----	Tx
(4,I)----	Highlighting	(10,I)---	Ty
(5,I)----	Detectability	(11,I)---	Head pointer
(6,I)----	Sx	(12,I)---	Tail pointer

Table 2 Segment information

後者は1次元配列でありシーケンシャルに用いている。

6. YN.CORE INPUT

入力機能は，応用プログラムのグラフィック・コマンドを解釈し，実行することが出来るインタラクティブ・ルーチンの集まりである．このコマンドは応用プログラム中で生成され，入力イベントを発生させる．応用プログラムはグラフィック・ディバイスからデータを受けとる．

6.1 CORE入力の概要

論理入力ディバイスは2つに分けられる．

- EVENT device: PICK,KEYBOARD,BUTTON,STROKE
- SAMPLED device: LOCATOR,VALUATOR

Table 3 Logical devices

EVENT deviceは応用プログラムにイベントが発生したことを知らせる．SAMPLED deviceは応用プログラムが情報を読み取ることが出来るディバイスである．EVENT deviceはSAMPLED deviceにはならないし，またその逆もありえない．

logical device	physical device(DSCAN接続)
PICK	Tablet
STROKE	〃
LOCATOR	〃
VALUATOR	〃
KEYBOARD	Keyboard
BUTTON	Button

Table 4 logical device and physical device

6.2 入力システム

応用プログラムが使える機能には以下のようなものがある．

```
% 1 Initialize and Enable
  2 Reading SAMPLED device
  3 EVENT Handling
  4 Associating device
  5 Accessing event report data
% 6 Synchronous input function
% 7 Device echoing
% 8 Setting input device characteristics
% 9 Inquiary
```

YN-COREは同期入力であるので、%印が機能となる．合計36個のルーチンである．

6.3 論理入力ディバイス用データ構造

これは3つの部分から成る．

(1) Input Capability Table ---入力全体の特徴を決定．

(2) Input Device Characteristics ---各ディバイスの特徴を指定．

(3) Event Report Buffer---入力情報を格納する．

これらはQ.B.より上部でアクセス出来る．

6.4 物理入力ディバイス用データ構造

(1) Event Report Buffer

(2) Datapacket

Datapacketは物理入力ディバイスからの情報を受け取り，物理入力ディバイスに必要な情報を送る為に使用される．Q.B.以下でアクセスできる．

6.5 YN-COREに於ける制限事項

(1) タブレットについているPen SwichとBUTTON NO.1は同一とする．

(2) 論理ディバイス番号は整数とし，1から始まる．

(3) イベントが起きるまで応用プログラムが待つ時間(Wall Clock)は現在考えていない．

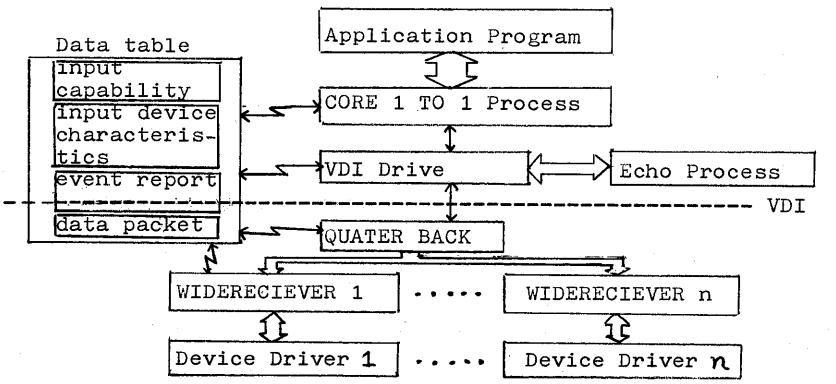

fig.7 入力機構

7. 問題点

7.1 Quater Back は概念上 above VDIとするか. (観点 1.)
below VDIとするか. (観点 2.)

(1) ディバイスが増設された際above VDI に対してシステム自体に変更を加えたくない. ---観点 2.
(2) ANSIに於いて、まだVDI提案が流動的. ---観点 2.
(3) 論理ディバイスと物理ディバイスの対応. ---観点 2.
(4) VDIの入口・出口を各々1つにすると構造がすっきりする. ---観点 1.

解決 以上の観点からQ.B.はbelow VDIとする.

7.2 セグメント・ファイルの位置をどこにするか.
——— above VDI / below VDI.

解決 VDI機能からみてもセグメントの保持はbelow VDI ですべきであるし、本来Device Driverが持つべきものであるが、メモリーのオーバーヘッド減らし、システムのバランスを良くする意味からabove VDI に設定する. 但し. buffered typeのディスプレイに対しては、その本来的意味からも、応答性を良くする為にも、その装置固有のセグメントを用いる.

7.3 入力機能にあらわれるTIME の処理.

解決 below VDI で行なう.

8. ディバイス・ドライバー, コンピュータ環境

8.1 資源

YN-COREを使用し応用プログラムを実行するには, TSSモードとBATCHモードの2種類があり、それぞれFORTRAN で16MBまでの(YN-COREシステムを含む)メモリを使用することができる.

また、入出力装置として複数の入出力装置, 汎用入出力装置, TSS回線装置を複数使用し, グラフィック・ディスプレイ, プロッタのそれぞれのもっている機能特性を十分にいかしたMulti surface 構成になっている.

8.2 ハードウェア構成

(1) グラフィック・ディスプレイ

表5に使用されるグラフィック装置の機能を示す.

ディバイス	回線装置	型式	解像度	色数	全インテリジェント	利用インテリジェント
グラフィカ MB501	TSS回線	ラスタースキャン	768×512	モノクロ	35	31
テクトロ 4010	〃	ストレージタイプ	1024×780	モノクロ	49	49
D-SCAN G2401	〃	ラスタースキャン	1024×1024	7色	96	65
グラフィカ M303	汎用入出力装置	ラスタースキャン	512×320	512色	21	21
岩通 XY-プロッタ	汎用入出力装置	プロッタ	∞×270%	3色	10	10

Table 5.

(2) データの通信型式

各ディバイスは、2,3種類のデータ送受信型式をもち、YN-COREに用いているディバイス・ドライバはそのうち1種類をサポートしている.

データ通信型式には大別して、文字列コードとエスケープ・コードを利用した

擬似ASCIIコード、特殊コードがあり、TSS回線につながるディバイスにはホスト計算機とのレスポンスを考慮し、擬似ASCIIコード方式を採用した。他のディバイスには特殊コードを送出する方法をとっている。

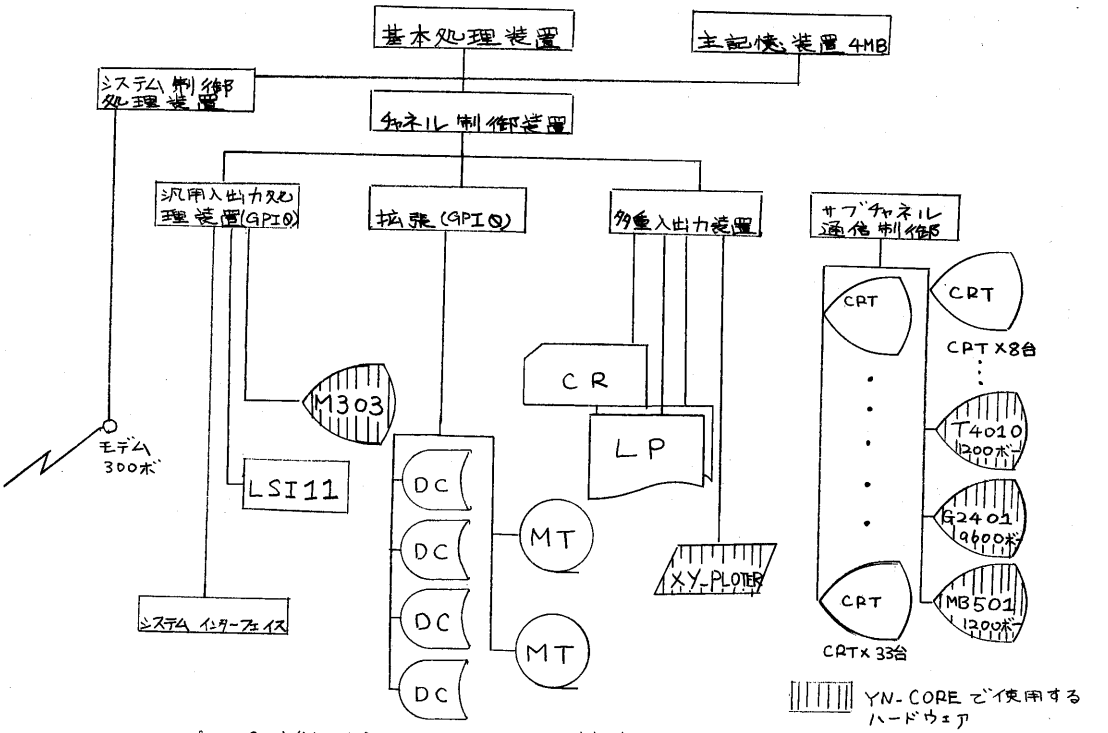

fig. 8 YN-CORE ハードウェア構成図

9. 謝辞

日本電気共通ソフトウェア開発部，及び㈱二精工舎の本研究への様々な支援に感謝します。

10. 参考文献

1) 穂坂 衛・木村 文彦，グラフィックスの標準化について，コンピュータ・グラフィックス研究会資料1，1981年6月

2) Status Report of the Graphic Standards Planning Comittee, Computer Graphics, SIGGRAPH-ACM, Vol. 13, no. 3, Aug. 1979

3) ANSI X3H33 Task Group POSITION PAPER, VIRTUAL DEVICE INTERFACE and VIRTUAL DEVICE METAFILE, X3H33 81-27R5, Dec. 28, 1981

4) Patricia Wenner, Design Document for the George Washington University Implementation of the 1979 GSPC CORE System, May 1980

5) Harold L. Curling, U.S. Air Force Institute of technology, Design of an Interactive Input Graphics System Based on the ACM CORE Standard, 1980