PHIGS89概説(II)

佐藤, 周行
九州大学大型計算機センター研究開発部

https://doi.org/10.15017/1470219

出版情報：九州大学大型計算機センター広報. 25 (6), pp.507-530, 1992-11-25. 九州大学大型計算機センター
バージョン：
権利関係：

PHIGS89 概説 (II)

佐藤周行*

1 はじめに

前回 ([2]) は PHIGS における図形の構成単位である出力基本要素について解説した。実際の意味のある「図形」(= モデル) は出力基本要素の単純、または複雑な組み合わせでできているわけである。今回はまずそのモデルをどう構成していくか、また編集していくかを説明する。

次に PHIGS における入出力の扱いについて説明する。対話的グラフィックスにおいては入出力をどう組み込むかということは基本的な問題である。ここではセンター 2 階に設置の PHIGS 専用 F6247 システムを対象にキーボードやタブレットがどう位置付けられ、扱われているかを解説する。最後に出力として一般のデータセットを指定する方法について簡単に説明する。

前回では具体的なプログラムがほとんどなく読者は実感がわかなかったかもしれない。今回はそれを反省し、付録に 小規模な具体例をソースプログラムの形であげている。(呼び出し形式についての説明を省いているサブルーチンがあるので細部を理解するにはマニュアルが必要であるが) 参考になればと思う。

2 ストラクチャ

前回 PHIGS における図形の構成単位である出力基本要素について解説した。PHIGS ではそれらを組み合わせてひとつの「図形」(= モデル) を構築するわけである。これら操作対象となる「図形」の単位をストラクチャ (structure) と呼ぶ。

モデルの編集はストラクチャの編集と言う形で実現される。

2.1 ストラクチャを開く / 閉じる (前回の復習)

前回説明した POPST はストラクチャを編集するために対象となるストラクチャを「開く」ということを意味する。ストラクチャの管理はプログラム側で ID(integer) をふって管理する。同時に開くことのできるストラクチャは 1 個である。

PCLST は 開いているストラクチャを「閉じる」サブルーチンである。

```
POPST(strid)
integer strid

PCLST
```

2.2 ストラクチャをポストする (前回の復習)

出力ワークステーションにストラクチャ出力を指示するには「ワークステーション `wkid`」と「ストラクチャ `strid`」を表示の優先度 `priority` とともに指定する。

平成 4 年 9 月 25 日 受理

*九州大学大型計算機センター研究開発部

```
PPOST(wkid, strid, priority)
integer wkid, strid
real priority
```

POST されているストラクチャ に対して編集その他を行なう時にそれが表示にどう反映されるかはワークステーションによって異なる。

F6247 ディスプレイシステムの場合即座に画面に反映されるようになっている。これを利用すると対話的に図形を変化させることが可能になる。

例 1 (F6247 でストラクチャの編集の結果を画面にすぐに反映させる)

```
          .
     POPPH(99,0)
C Open F6247.
      POPWK(wkid, 1,1)
      POPST(strid)
C
C Post the structure strid to F6247.
C
      PPOST(wkid, strid, 1.0)
C
C NOW the structure strid is open, so the result of editing is
C reflected on the screen.
          .
          .
      PCLST
          .
```

具体例が プログラム例 A, B に与えられているので参照すると良い。

「ポスト」された状態のストラクチャは PUPOST (Phigs **UnPOST**) によりポスト状態を解消できる。

```
PUPOST(wkid, strid)
integer wkid,strid
```

2.3　ストラクチャ に設定できる要素

ストラクチャに書き込むことのできる要素は以下のものである。

- 出力基本要素
- 属性
- 座標変換要素
- Structure 実行要素
- ラベル

以下、各々について具体的に説明する。

2.3.1　出力基本要素

2.3.2　属性

これら 2 つについては前回解説したので省略する。

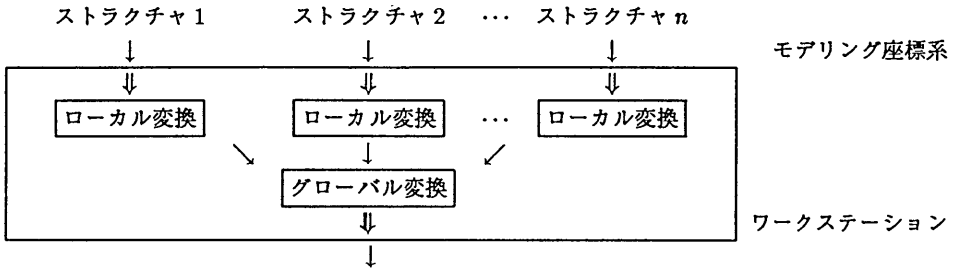


図 1: 座標系の変換

2.3.3 座標変換要素

これまでも出力基本要素を指定する際には 座標の指定が必要だった。この座標 を特に モデリング座標系による座標という。図形は ストラクチャにローカルなモデリング座標系によって記述され、最後にグローバルなモデリング座標の変換を受けて表示される (図 1参照)。

ストラクチャ 生成にあたっては各要素を並べていくわけだが、その座標系を途中で変換していくことができる。これをするのが座標変換要素である。座標変換要素は 3 次元デカルト座標系上でのアフィン変換の 4 × 4 の表現行列で指定する。

なお、2 次元の場合の表現行列は 3 × 3 になるが本質は同じなので本稿では解説を省略し、以後 3 次元の変換のみを扱うことにする。

PSLMT3 (Phigs Set Local transformation MaTrix 3)

```
PSLMT3(matrix, mode)
real matrix(4,4)
integer mode
```

mode に応じてローカル変換マトリックス XFRMT は以下のように変換される

mode	変換	意味
0	XFRMT ⟵ XFRMT × matrix	後から適用
1	XFRMT ⟵ matrix × XFRMT	前に適用
2	XFRMT ⟵ matrix	matrix に置換

ローカル変換マトリックスのスコープは編集対象になっているストラクチャ内である。全ストラクチャにわたって変換を施す場合は以下のものを用いる。

PSGMT3 (Phigs Set Global transformtaion MaTrix 3)

```
PSGMT3(matrix, mode)
real matrix(4,4)
integer mode
```

例として座標変換を施しながら線を引いていって☆型を作っていくストラクチャ をあげる。

例 2 (☆をつくる) ☆型を作る方法のうち、一番安易なのが以下の方法である。

1. 原点から一方向に線分を引く。

2. 座標系を $6/5\pi$ 回転させてさらに原点を線分のもう一つの端に持っていく。
3. 上のプロセスを 5 回繰り返す。

さて、プログラム A から☆を定義しているストラクチャ star を抜き出してこよう。A はデモの見栄えその他を気にしていろいろ飾りがついているが本質は以下の部分のみである。xfrmt は「座標系を $6/5\pi$ 回転させてさらに原点を線分のもう一つの端に持っていく」変換に対応する 4×4 の行列になっている。読者は下のプログラムが上の方針を忠実に反映していることを読みとられたい。なお、プログラム A の実行結果を図 A にのせる。座標系の変化の様子を読みとることができるだろう。

```
      real ulinex(2),twozero(2)
      data ulinex/0.0,2.0/,twozero/0.0,0.0/
      real xfrmt(4,4)
            .
            .
      call pbltm3(0.0,0.0,0.0,2.0,0.0,0.0,0.0,0.0,0.0,
     -           1.2 * PI, 1.0,1.0,1.0,errind,xfrmt)
            .
C draw a star
      call popst(star)
            .
C draw a unit line, transform the coordinate system,
C draw a unit line again, ...
      do 100 i=1,5
            .
          call ppl3(2,ulinex,twozero,twozero)
            .
          call pslmt3(xfrmt,after)
  100 continue
            .
      call pclst
```

アフィン変換に対しては特に表1 に示すようなユーティリティを用意し、変換行列の計算に便宜をはかっている。一例として 付録で用いているルーチン PBLTM3 の呼び出し形式をあげる。その他のルーチンについてははマニュアルを参照すること。

PBLTM3(Phigs BuiLd Transformation Matrix)

```
PBLTM3(x0,y0,z0, dx,dy,dz, thetax,thetay,thetaz, fx,fy,fz,err, xfrmt)
real x0,y0,z0, dx,dy,dz, thetax,thetay,thetaz, fx,fy,fz
integer err
real xfrmt(4,4)
```

このサブルーチンは xfrmt に以下の変換を施す行列を返す。

1. (x0,y0,z0) を原点に
2. 各軸 (x,y,z) に対して (fx,fy,fz) だけ拡大 / 縮小し、
3. 原点を通り各軸 (x,y,z) に平行な線を中心にしてそれぞれ (thetax,thetay,thetaz)(ラジアン) だけ回転し
4. さらに (dx,dy,dz) だけ平行移動する。

PTR3	平行移動 (TRanslation)
PSC3	拡大縮小 (SCaling)
PROX, PROY, PROZ	回転 (ROtation X,Y,Z)
PBLTM3	上の 3 種類の変換の組み合わせ (新規作成)
PCOTM3	上の 3 種類の変換の組み合わせ (合成)
PCOM3	4 × 4 行列の積
PTP3	変換行列を座標点に適用

表 1: 変換行列の計算用のユーティリティ

2.3.4 ストラクチャ実行要素

他のストラクチャもストラクチャの構成要素になる。ストラクチャを呼び出すには PEXST を用いる。

PEXST(Phigs EXecute STructure)

```
PEXST(strid)
integer strid
```

この場合、座標系はストラクチャのモデリング座標系を、属性もストラクチャの属性を引き継ぐ。

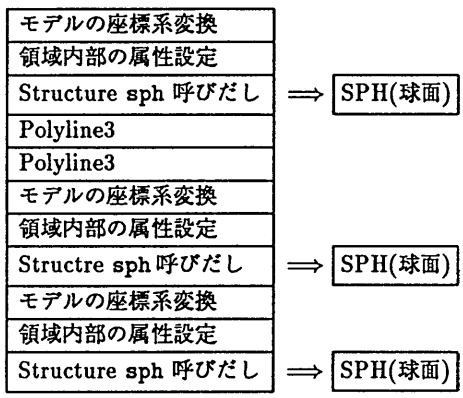
例として球面を 3 個並べる水分子模型を作るプログラム B を見てみよう。ここではストラクチャ sph において球面を定義し、ストラクチャ water においてその球面を 3 回呼び出している。ストラクチャ water の該当部分を切り出して見よう。

例 3 (ストラクチャ water のごく単純な定義) プログラム Bの定義 とはラベルの部分を除いて一致する。

```
      call pbltm3(0.0,0.0,0.0,0.5,0.5,0.5,1.0,0.0,1.0,0.1,0.1,0.1,
     -            errind,xfrmt)
      call popst(2)
C sphere 1
      call pslmt3(xfrmt,2)
      call psici(1)
      call pexst(1)
C bond 1
      call ppl3(2,pplx,pply,pplz)
C bond 2
      pply(2) = -1.0
      call ppl3(2,pplx,pply,pplz)
C sphere 2
      call pbltm3(0.0,0.0,0.0,pplx(2),1.0,0.0,0.0,0.0,0.0,0.0,0.66,0.66,0.66,
     -            errind,xfrmt)
      call pslmt3(xfrmt,0)
      call psici(2)
      call pexst(1)
C sphere 3
      call pbltm3(0.0,0.0,0.0,0.0,-3.0,0.0,0.0,0.0,0.0,0.0,
     -            1.0,1.0,1.0,errind,xfrmt)
```

```
call pslmt3(xfrmt,0)
call psici(3)
call pexst(1)
call pclst
```

これを少し抽象的にまとめると次のようになる。



呼び出されたストラクチャの座標系と属性がそれぞれ呼び出す側のそれを引き継いでいることに注意すること。

このようにストラクチャは一般に互いの「呼び出し」によってストラクチャのネットワークを作っている。このネットワークを管理するのが CSS である。

2.3.5　ラベル

ラベルはストラクチャの編集の時に効果を発揮する。ストラクチャ内のポインタの役割と併せて 3.3で説明する。

3　ストラクチャ の編集

3.1　ストラクチャ の編集とは

今までも「ストラクチャの編集」という言葉を何気なく用いてきたが、ここで「編集」ということをまとめてみよう。

PHIGS ではストラクチャの編集機能として以下のものを用意している。

1. 編集モードの設定
2. ポインタの設定
3. ラベルづけ
4. 要素の挿入 / 置換 / 削除
5. ストラクチャ間の要素コピー
6. ストラクチャ 削除
7. ストラクチャ 改名
8. その他

ここでは前節で作った水分子の模型を対象に上の機能のうち主なものをざっと眺めることにしよう。

ストラクチャが開かれている時、その状態はそのストラクチャ内の要素の並びと、そのうちのどれを「指して」いるかを示すポインタで記述される。例えば、水分子模型を定義するストラクチャを定義し終って閉じる直前、ストラクチャは前節の要素の並び、ポインタは最後の要素を指している。

例 4 (ストラクチャ water を閉じる直前の状態)

	1	モデルの座標系変換		
	2	領域内部の属性設定		
	3	Structure sph 呼びだし	⟹	SPH(球面)
	4	Polyline3		
	5	Polyline3		
	6	モデルの座標系変換		
	7	領域内部の属性設定		
	8	Structure sph 呼びだし	⟹	SPH(球面)
	9	モデルの座標系変換		
	10	領域内部の属性設定		
⟹	11	Structure sph 呼びだし	⟹	SPH(球面)

3.2 編集モードの設定

PSEDM(Phigs Set EDit Mode)

```
PSEDM(edmode)
integer edmode
```

編集はポインタの指す要素を対象にして行なう。要素を書き込む場合、ポインタの指す要素の「後ろに挿入 (挿入モード =0)」するか、「置換 (置換モード =1)」するかのモード選択を行なう。デフォルトは挿入モードである。

3.3 ラベルとポインタの設定

ストラクチャを開いた直後、ポインタはストラクチャの最後の要素を指す。PHIGS では最初の要素に番号 1 を割り振る。ポインタ 0 は先頭の要素の前を指すことになる。

PSEP(Phigs Set Element Pointer)

```
PSEP(pointer)
integer pointer
```

call psep(0) はストラクチャの先頭にポインタを設定することを意味することになる。PSEP を使って良いのはこの場合だけである。後は (後述の) ラベルとそこからのオフセットによってポインタを動かすべきである。

さて、ストラクチャにはラベルを書き込むことができる。ラベルが指定されていれば要素はラベルとそこからのオフセットによって指定できる。PSEP のように絶対番値を指定すると、ポインタの管理が大変になる。

PLB(Phigs LaBel)

```
PLB(label)
integer label
```

ラベルを書き込む。

PSEPLB(Phigs Set Element Pointer at LaBel)

```
PSEPLB(label)
integer label
```

ラベル label まで飛ぶ。ただし wrap-around しない。

POSEP(Phigs OffSet Element Pointer)

```
POSEP(offset)
integer offset
```

現在のポインタ位置から offset ぶんだけポインタをずらす。

さて、水分子模型を定義し直すことにする。O, H, ボンドにそれぞれラベルを設定することにする。

結果のストラクチャは以下の通り。これはプログラム B 内のストラクチャ water を正確に反映している。

モデルの座標系変換		
領域内部の属性設定		
ラベル (spho)		
Structure sph 呼びだし	⟹	SPH(球面)
Polyline3		
Polyline3		
モデルの座標系変換		
領域内部の属性設定		
ラベル (sphh0)		
Structure sph 呼びだし	⟹	SPH(球面)
モデルの座標系変換		
領域内部の属性設定		
ラベル (sphh1)		
Structure sph 呼び出し	⟹	SPH(球面)

3.4　要素の挿入 / 置換 / 削除

水分子模型を例にとって考えよう。ここでは水分子模型を開いて、酸素原子部分に注釈をいれることを考える。

以下のプログラム断片を考える。まずストラクチャ water を開く。ストラクチャを開いた直後の状態は図 2 -1 のとおり。

次にポインタを先頭に持ってきて (PSEP(0))、ラベル spho + 1 までポインタを移す (PSEPLB(spho) + POSEP(1))。さらに注釈文字列を挿入する (PATR3(座標, 'OXYGEN'))。ここまで実行した時のストラクチャの状態は図 2 -2 のようになる。

```
POPST(water)
PSEP(0)
PSEPLB(spho)
POSEP(1)
PSTXCI(white)
PATR3(座標、 'OXYGEN')
```

さて、プログラム B では対話的に注釈文字列の変更ができるようになっている。これに対応するプログラムの断片は以下のようになる[1]。

まず、編集モードを置換モードに変換する (PSEDM(subst))。そして PATR3(座標、 STR) を実行するとポインタ部分の要素 (注釈文字列) が PATR3(座標、 STR) の実行結果で置換される。ここまで実行した時のストラクチャの状態は図 2 -3 のようになる。

```
     PSEDM(subst)
     文字列入力の設定
1000 STR = 入力文字列
     PATR3(座標、 STR)
     goto 1000
```

最後に注釈文字列を削除することにする。サブルーチン PDEL(Phigs Delete ELement) がこれに対応する。実行後、ポインタは削除対象の要素の一つ前を指す。

PDEL

最終的には図 2 -4 のようになる。コードとしては B の後半部分がこれらに対応する。

3.5 その他

その他、以下に示すようなサブルーチンが用意されている。

3.5.1 ストラクチャ要素のコピー

ストラクチャを指定してその中の要素をコピーする。コピー後のポインタは複写されたもののうち、最後の要素に移る。

PCELST(Phigs Copy all ELements from STructure)

3.5.2 ストラクチャ要素の範囲指定削除

二つのポインタ、またはラベルを指定してその範囲にある要素を全部削除する。

PDELRA(Phigs Delete ELement RAnge)

PDELLB(Phigs Delete ELements between LaBels)

3.5.3 ストラクチャ全要素削除

PEMST(Phigs EMpty STructre)

3.6 ストラクチャを対象にした編集

今まではストラクチャの要素を対象にした編集のみを説明してきたが、CSS つまりストラクチャのネットワークを対象にした編集もできる。

3.6.1 ストラクチャ削除

PDST(Phigs Delete STructure)

ストラクチャを指定して削除する。

[1]入力部分は 4節 で説明するのでここでは触れない。

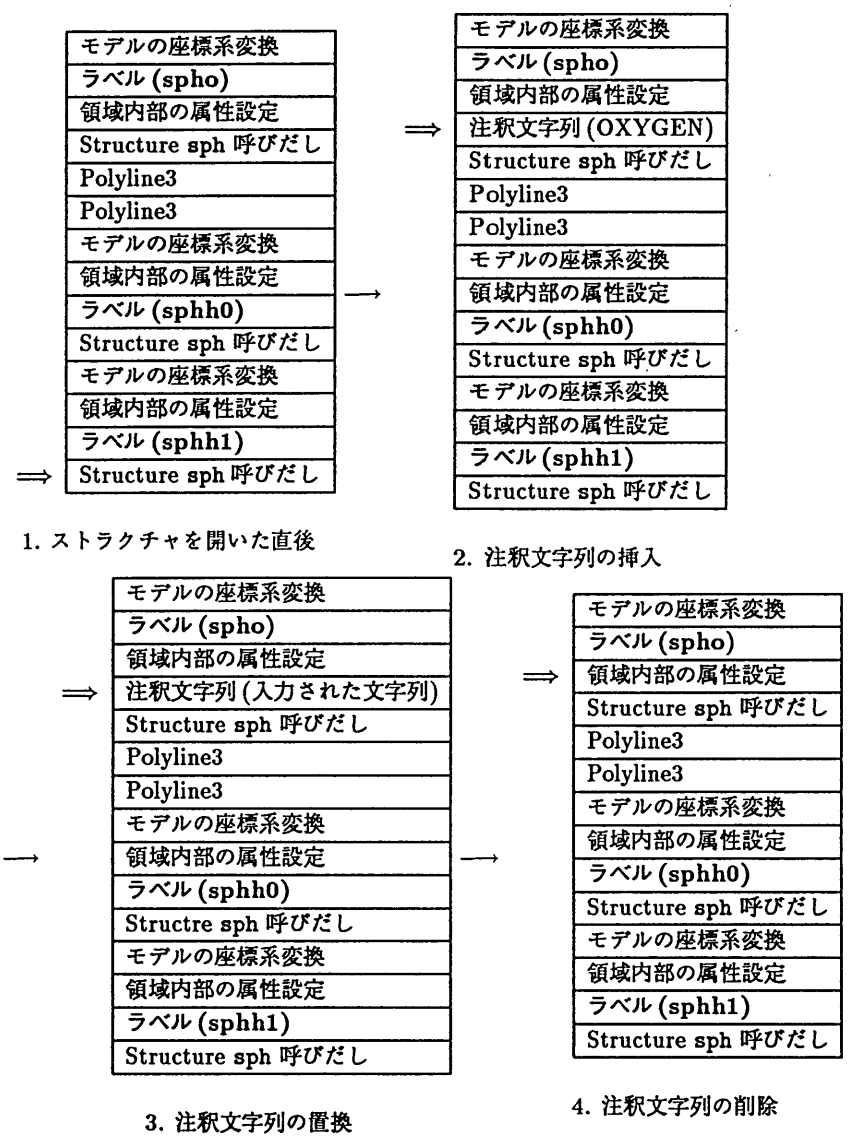


図 2: ストラクチャの編集例

3.6.2 ストラクチャネットワーク削除

PDSN(Phigs Delete Structure Network)

ストラクチャを指定して削除する。PDST との違いは water を例にとると、pdst(water) で削除されるのは water のみだが、pdsn(water) とすると、water の中から呼び出されているストラクチャ sph も同時に削除されることである。

3.6.3 全ストラクチャ削除

PDAS(Phigs Delete All Structres)

それぞれについての詳細はマニュアルを参照すること。

4 ワークステーションとデータの入出力

PHIGS で言う「ワークステーション」とは 物理デバイスを抽象化したものである。PHIGS は入出力をこの抽象化された「ワークステーション」を対象として行なう。それを現実の図形入出力に対応させるのは PHIGS システムの実現の問題である。ワークステーションには大別して次の 4 種類がある。

入出力ワークステーション データの入力と図形出力を併せてできるデバイス。 センター設置の F6247 ディスプレイシステムはこれに相当する。ディスプレイに図形出力ができ、なおかつキーボード、タブレットを通じてデータの入力ができる。

出力ワークステーション 図形出力のみできるもの。例えばプリンタはこれに対応する。 九大センター MSP の PHIGS ではこれはサポートされていない。

入力ワークステーション データ入力のみできるもの。デジタイザなど。九大センター MSP の PHIGS ではこれはサポートされていない。

メタファイル入力 / 出力 データセット (メタファイル) を通じて図形データの入出力を 行なうもの。

九大センターでは PHIGS 専用端末として 入出力ワークステーションである F6247 ディスプレイシステム、さらにメタファイル入力 / 出力 をサポートしている。

以下、具体的に F6247 ディスプレイシステムとメタファイルについての解説をする。

5 F6247 ディスプレイシステム での入出力

今までは図形出力のみの説明をしてきたのでここでは F6247 システムのもう一つの機能であるデータ入力の方法のみを説明する。

前回すでに触れたが、 F6247 ディスプレイシステムを利用するにはプログラム側で以下のようにする。

```
POPWK(wkid, 1, 1)
```

上のように指定しておけば phigs89 コマンドが F6247 システムとのインターフェイスを取ってくれる。

5.1 データ入力

センター設置の F6247 システムでは 入力デバイスとしてキーボードとタブレットを用意している。

PHIGS では入力を 表 2 の 6 種類に分けている。さらに入力モードとして要求入力、サンプリング入力、イベント入力の 3 種類がある。データ入力をするためには装置の 1. 初期化と 2. 入力モード選択をしなければならない。以下では文字列入力を例にとって具体的に説明する。さらに実際のプログラム B を必要に応じて引用することにする。

残りの入力については省略するが骨格は変わらないので類推は容易にできるはずである。

入力の種類	入力値	初期化ルーチン	装置番号(対応する入力装置)
位置入力 (locator)	座標値	PINLC3,PINLC	1 (タブレット)
点列入力 (stroke)	座標値列	PINSK3,PINSK	1 (タブレット)
実数値入力 (valuator)	実数値	PINVL3,PINVL	1 (キーボード)
選択値入力 (choice)	選択値	PINCH3, PINCH	1 (PF キー) 3 (タブレット)
ピック入力 (pick)	ピック状態	PINPK3, PINPK	1 (タブレット)
文字列入力 (string)	文字列	PINST3, PINST	1 (キーボード)

表 2: 入力の種類と対応する入力装置

5.2　位置入力の初期化

F6247 システムでは以下のようにする。

PINST3(Phigs INitialize STring 3)

```
pinst3(wkid, stdnr, lstr, initstr, pet, evol, ldr, datrec)
integer wkid,stdnr,lstr
character*(*) str
integer pet
real evol(6)
integer ldr
character*80 datrec(ldr)
```

wkid には使用中のワークステーションの ID を指定する。stdnr はキーボードに対応する装置番号 (F6247 の場合 1) を指定する。str は初期文字列を長さ (lstr) とともに指定する。pet はプロンプトエコータイプを指定する。この場合 1 に固定すること。evol はエコーボリュームを指定する。エコーボリュームはエコーの表示領域を装置座標系で指定する。この説明は座標系の理解が必要なのでとりあえず詳細は省略する。次回以降 座標系の説明をしたあとに必要に応じて説明しよう。datrec は上で指定し切れない細かい指定をするために確保されている。詳細はマニュアル参照。ただし、具体例を載せるのでそれを真似すれば大体のところは間に合う。

ldr, datrec はユーティリティサブルーチン PPREC を使って作る。

プログラム例 B で該当する部分を抜きだそう。

```
      integer f6247,keyb
      parameter(f6247=1,keyb=1)
            .
            .
      character*80 chars
      data chars/'SOME MOLECULE'/
      integer ia(2),ldr,
      character*80 datrec(1)
      data ia/80,1/
C
      real evol(6)
      data evol/0.0,1023.0,0.0,50.0,0.0,32767.0/
            .
            .
```

```
      call pprec(2,ia,0,0,0,0,0,1,errind,ldr,datrec)
      call pinst3(f6247,keyb,13,chars,
     -            1,evol,ldr, datrec)
C
```

さて初期化は済んだ。

5.3 入力モードの設定

次にすべきは入力モードの設定である。モード設定はサブルーチン PSSTM(Phigs Set STring Mode) で行なう。

```
PSSTM(wkid, stdnr, mode, esw)
integer wkid,stdnr,mode,esw
```

入力モードには 要求入力 (0)、サンプリング (1)、イベント入力 (2) の 3 種類があり、 mode で指定する。esw は入力をエコーするか (1) しないか (0) を指定する。
B において対応する部分を抜き出す。

```
      .
      call psstm(f6247,keyb,pevent,pecho)
      .
```

5.4 データ入力

入力モードによって変化する。入力モードが要求入力かサンプリング入力の場合は以下のサブルーチンを呼び出すだけである。

1. 要求入力の場合。

   ```
   PRQST(wkid, stdnr, stat, lostr,str)
   ```

   PRQST の実行によりプログラムは入力待ちになる。返ってきた時に stat に入力状態 (中断 / 完了) が、str に入力文字列が入る。

   プログラム例 A は選択値入力の例になっている。

2. サンプリング入力の場合。

   ```
   PSMST(wkid, stdnr, lostr,str)
   ```

   str に入力文字列が入る。

イベント入力の場合は以下のようにする。
骨格は

1. イベント待ち状態に入る。

   ```
   PWAIT(timeout, wkid, icl, idnr)
   real timeout
   integer wkid, icl, idnr
   ```

   timeout はタイムアウトの間隔を指定する。

icl	意味	入力関数
0	イベントなし	
1	位置入力	PGTLC3, PGTLC
2	点列入力	PGTSK3, PGTSK
3	実数値入力	PGTVL
4	選択値入力	PGTCH
5	ピック入力	PGTPK
6	文字列入力	PGTST

表 3: イベントの種類と対応する入力関数

2. イベント通知を受けとる。PWAIT から戻ってきた時に icl にイベントの種類、 idnr にそのイベントが発生した装置番号が入っている。
3. イベントの種類に応じて対応する入力サブルーチンを起動する。具体的には表 3 参照

プログラム例 B で対応する部分を抜き出す。

```
          .
1000 call pwait(30.0, f6247, icl, idnr)
     if (icl .eq. pnclas) goto 2000
     call pgtst(lostr,str)
          .
```

6　メタファイルを用いた入出力

センター提供の PHIGS では F6247 システムの他にメタファイルという形でデータセットを通じて PHIGS のデータを読み書きできるようになっている。

6.1　メタファイルへの出力

メタファイル出力をするためには まずTSS 側で以下のような環境設定をする。

6.1.1　ワークステーション型定義ファイルの用意

ワークステーションの型を指定する。まず FB 形式、レコード長 80 バイト、ブロック長 3120 バイトのデータセットを用意する。名前は仮に WSK.DATA とでもしておこう。内容は次のようにする。

```
 READY
list wsk.data
 A79999A.WSK.DATA
      WST0001=3;
      1:30402000;
      2:10100000;
      3:10002000;

 END OF DATA
 READY
```

WST0001=3 の '3' は定義された型の総数を表す。POPWK の第 3 引数で指定する「ワークステーション型」はこのデータセットで記述する。例えばこの環境下で型として '3' を指定した時、これは '10002000' のことである。

これ以外の指定をする時にはマジックナンバーの意味を知る必要がある。本稿では省略する。知りたい読者はマニュアルを参照すること。

6.1.2 データセットの用意

次にメタファイルに対応させるデータセットを用意する。形式は以下の通り。

データセット名	任意			
データセット編成	PS or PO			
レコード形式	F	FB	V	VB
レコード長 (LRECL)	≥ 80			
ブロック長	LRECL	LRECL の倍数	LRECL + 4	(LRECL の倍数) + 4
DD 名	FTxxF001			

PHIGS89 コマンド実行前に次のようにして環境設定をする。ここではメタファイル用のデータセット形式は FB ということにする。

```
 READY
alloc f(phgenv) da(wsk.data)
 READY
attr phgatr block(3120) lrecl(80) rec(F B) ds(PS)
 READY
alloc f(ft02f001) da(meta.data) new using(phgatr)
 READY
```

6.1.3 プログラム側での用意

PHIGS のプログラム側で以下のようなプログラムを書いたとする。

```
      .
    call popwk(meta, 2, 3)
      .
```

ここでは POPWK の第 2 引数と第 3 引数が以下の意味を持っている。

1. 第 2 引数の '2' は メタファイルとして割り当てたデータセットの機番 FT02F001 の '2' であり、
2. 第 3 引数の '3' は wsk.data に記述されたワークステーション型 '3' のことである。

ここまでの環境設定が順調にいけばプログラムで ワークステーション meta へ「ポスト」することが MSP の世界で データセット meta.data へ書き込むことに対応することになる。

6.2 メタファイルからの入力

メタファイルを入力として用いる場合のワークステーション型は '10100000' に対応すること。前述の環境では以下のようにする。あとの設定は出力と同一。

```
      .
    call popwk(meta, 2, 2)
      .
```

メタファイルから各データ(項目)を読み込んで解釈するために以下のサブルーチンが提供されている。詳細は省略する。マニュアル参照。

PGTITM	項目を取り出す(ポインタそのまま)
PRDITM	項目を読み出す(ポインタを次の項目へ)
PIITM	項目を解釈する

7　プリンタへの出力

センターのPHIGSではプリンタに対応するワークステーションが提供されていないので印刷はPHIGSの枠内ではできない。ただし、以下の方法で「プリンタへの出力」ができる。

センター F6247 附属のカラーハードコピー装置を利用する

これが一番簡単である。センター2階に来ればすべてが理解できる。

GKS がプリンタ出力可能であることを利用する

PHIGS と異なり、GKS ではプリンタ出力をサポートしている。しかも 都合のよいことに メタファイルの形式に互換性がある。従って、PHIGS で作ったメタファイルを GKS で読みだし、プリンタへ出力することができる。詳細は [1] を参照。

付録Cにメタファイルを読み込んで出力するプログラムのプロトタイプを載せるので参照されたい。なおこれは本センター山嵜信広氏の提供による。

8　おわりに

今回も複数の話題を駆け足で説明する形になってしまった。プログラミングに重点を置いているので説明がどうしても ad-hoc になり、体系的な説明を求める読者にはわかりづらいものになったかもしれない。次回は座標系の話をする予定であるが、一段落したら説明を整理することを試みることにしよう。

参考文献

[1] 橋倉他,「グラフィックツールによる図形出力ガイド」,広報 Vol. 23, No.4, 1990, pp. 323–386.

[2] 佐藤周行,「PHIGS89 概説 (II)」,広報 Vol. 25, No.5, 1992, pp. 393–404.

付録: プログラム例

A ☆型を描画する

以下のプログラムは☆型を描画するものである。F6247上で実行する。絵が出て来たら5回PFキーかENTERキーを押すと、座標系を変換しながら☆を書いていく様子が観察できる(図A参照)。

特徴は

1. モデリング座標系の変換の例になっている(例2参照)
2. 要求入力モードでのキーボードからの入力の例になっている(5.4参照)
3. 図形を対話的に編集する例になっている(2.2参照)

ところである。

```
      program stargrm
C parameter
      integer pfkey,f6247,axis,star
      integer replace,after,red,yellow
      real PI
      parameter(pfkey=1,f6247=1,axis=1,star=2,
     -          replace=2,after=0,
     -          red=2,yellow=3,PI=3.1415926535)
C echo volume etc.
C -- magic data for the PF keys in PHIGS
      real evol(6)
      data evol/0.0,1023.0,0.0,1023.0,0.0,32767.0/
      integer ia(1)
      data ia/0/
      integer errind,ldr,stat,chnr
      character*80 datrec(2)
C coordinates
      real xaxx(2),yaxy(2),ulinex(2)
      real xarx(3),xary(3),twozero(2)
      data xaxx/-1.0,1.0/,yaxy/-1.0,1.0/
      data ulinex/0.0,2.0/
      data xarx/-0.05,0.0,0.05/,xary/0.9,1.0,0.9/
      data twozero/0.0,0.0/
C
C transformation matrix
C
      real xfrmt(4,4),xfrmti(4,4)
C
C open PHIGS and F6247
C
      call popph(99,0)
      call popwk(f6247,1,1)
C
C define axis
C
      call popst(axis)
      call ppl3(2,xaxx,twozero,twozero)
      call ppl3(2,twozero,yaxy,twozero)
      call ppl3(3,xarx,xary,twozero)
      call ppl3(3,xary,xarx,twozero)
      call psatch(0.03)
      call patr3(1.0,0.0,0.0,0.03,0.03,0.03,'X')
      call patr3(0.0,1.0,0.0,0.03,0.03,0.03,'Y')
      call patr3(0.0,0.0,0.0,0.03,0.03,0.03,'O')
      call pclst
C
C end of def. of axis
C
      call pbltm3(0.0,0.0,0.0,0.4,0.7,0.0,
     -      0.0,0.0,0.0,0.15,0.15,1.0,errind,xfrmti)
      call pbltm3(0.0,0.0,0.0,2.0,0.0,0.0,0.0,0.0,0.0,
     -          1.2 * PI, 1.0,1.0,1.0,errind,xfrmt)
C draw a star
      call popst(star)
      call ppost(f6247,star,1.0)
C
C star is under posting,
C so we can draw star interactively.
C
      call pprec(1 ,ia,0,0,0,0,0,2,errind,ldr,datrec)
      call pinch3(f6247,pfkey,1,1,1,evol,ldr,datrec)
      call pslmt3(xfrmti,replace)
C
C draw a unit line, transform the coordinate system,
C draw a unit line again, ...
C
      do 100 i=1,5
          call pslwsc(5.0)
          call psplci(red)
          call ppl3(2,ulinex,twozero,twozero)
          call pslwsc(1.0)
          call psplci(yellow)
          call pstxci(yellow)
          call pexst(axis)
          call prqch(f6247,pfkey,stat,chnr)
          call pdel
          call pslmt3(xfrmt,after)
  100 continue
C
      call prqch(f6247,pfkey,stat,chnr)
      call pclst
C
C epilogue
C
      call pclwk(f6247)
      call pclph
      stop
      end
```

B　水分子模型を作る

以下のプログラムは水分子の模型を作成、描画するものである。F6247 上で実行する。絵が出て来たら文字列の入力待ちになり、入力された文字列が注釈文字列として画面に反映される。なお、分子の大きさの比は見やすさのためにデフォルメしてある (図 B 参照)。

特徴は

1. 自明でないストラクチャネットワークの例になっている (2.3.4参照)
2. ストラクチャ編集の例になっている (3 参照)
3. イベント入力モードでのキーボードからの入力の例になっている (5.4参照)

ところである。

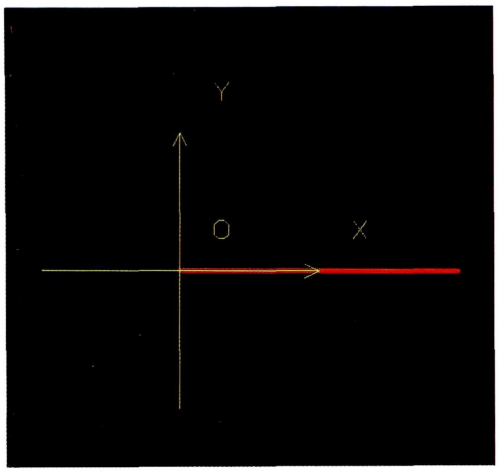
```
      PROGRAM sphere
C     parameters
      integer water,sph,spho,sphh0,sphh1
      integer f6247,keyb,subst
      parameter(sph=1,water=2,
     -          spho=1,sphh0=2,sphh1=3)
      parameter(f6247=1,keyb=1,subst=1)
      integer replace, after, yellow, red, white
      parameter(replace=2,after=0,
     -          yellow=3,red=2,white=7)
      real pi
      parameter(pi=3.1415926535)
      integer pnclas,pecho,pevent
      parameter(pnclas=0,pecho=1,pevent=2)
C
C magic data for string input
C
      character*80 chars
      data chars/'SOME MOLECULE'/
      integer ldr, ia(2)
      character*80 datrec(1)
      integer idnr,icl
      integer lostr
      character*80 str
C
      real evol(6)
      data evol/0.0,1023.0,0.0,50.0,0.0,32767.0/
      data ia/80,1/
C
      real fax(1000),fay(1000),faz(1000)
      real ej,ejp,ei,eip
C
C for bond
C
      real pplx(2),pply(2),pplz(2)
      integer ixa(250)
      integer npl
C
C n --- number of partitions
C the larger n, the finer sphere you get
C
      integer n
      real theta
      real xfrmt(4,4)
      integer errind
      n = 10
      theta = pi / n
C
      npl = n * n * 2
      do 100 i = 0, 2 * n -1
        do 100 j = 0, n-1
          ej = j * theta
          ejp= (j+1) * theta
          ei = i * theta
          eip = (i+1) * theta
C
C fax --X-coordinate of a <unit> sphere
          fax(4*(i*n+j) + 1) = sin(ej) * cos(ei)
          fax(4*(i*n+j) + 2) = sin(ejp)* cos(ei)
          fax(4*(i*n+j) + 3) = sin(ejp)* cos(eip)
          fax(4*(i*n+j) + 4) = sin(ej) * cos(eip)
C fay --Y-coordinate of a <unit> shpere
          fay(4*(i*n+j) + 1) = sin(ej) * sin(ei)
          fay(4*(i*n+j) + 2) = sin(ejp)* sin(ei)
          fay(4*(i*n+j) + 3) = sin(ejp)* sin(eip)
          fay(4*(i*n+j) + 4) = sin(ej) * sin(eip)
C faz --Z-coordinate of a <unit> sphere
          faz(4*(i*n+j) + 1) = cos(ej)
          faz(4*(i*n+j) + 2) = cos(ejp)
          faz(4*(i*n+j) + 3) = cos(ejp)
          faz(4*(i*n+j) + 4) = cos(ej)
  100 continue
C
C
      do 200 i = 1,npl
         ixa(i) = 4 * i
  200 continue
C
C open phigs
C
      call popph(99,0)
      call popwk(f6247,1,1)
C
C define a unit sphere
C
      call popst(sph)
      call pfas3(npl,ixa,fax,fay,faz)
      call pclst
C
C end of def. of a unit shpere
C
      pplx(1) = 0.0
```

```
      pply(1) = 0.0
      pplz(1) = 0.0
      pplx(2) = sqrn(3.0)
      pply(2) = 1.0
      pplz(2) = 0.0
C
C open structure water
C
      call pbltm3(0.0,0.0,0.0,0.5,0.5,0.5,
     -        1.0,0.0,0.6,0.1,0.1,0.1,errind,xfrmt)
      call popst(water)
C
C SPHere Oxygen
C
C because the structre sph is a unit shere,
C you must set the size and the coordinates
C manually if you put sph in any size
C and in any position.
C
      call pslmt3(xfrmt,replace)
      call plb(spho)
      call psici(yellow)
      call pexst(sph)
C
C bond 1
C
      call pslwsc(5.0)
      call psplci(white)
      call ppl3(2,pplx,pply,pplz)
C
C bond 2
C
      pply(2) = -1.0
      call ppl3(2,pplx,pply,pplz)
C
C SPHere Hydrogen 0
C
      call pbltm3(0.0,0.0,0.0,sqrt(3.0),1.0,0.0,
     -      0.0,0.0,0.0,0.66,0.66,0.66,errind,xfrmt)
      call pslmt3(xfrmt,after)
      call plb(sphh0)
      call psici(red)
      call pexst(sph)
C
C SPHere Hydrogen 1
C
      call pbltm3(0.0,0.0,0.0,
     -        0.0,sqrt(3.0)/0.66,0.0,0.0,0.0,0.0,
     -        1.0,1.0,1.0,errind,xfrmt)
      call pslmt3(xfrmt,after)
      call plb(sphh1)
      call psici(red)
      call pexst(sph)
      call pclst
C
C post spheres
C
      call pprec(2,ia,0,0,0,0,0,1,errind,ldr,datrec)
      call pinst3(f6247,keyb,13,'SOME MOLECULE',
     -            1,evol,ldr, datrec)
C
C an example of input
C
      call popst(water)
      call ppost(f6247,water,1.0)
      call psep(0)
      call pseplb(spho)
      call posep(1)
      call pstxci(white)
C
C initial annotation text is OXYGEN
C
      call patr3(0.0,0.0,0.0,-0.1,0.1,0.2, 'OXYGEN')
      call psedm(subst)
      call psstm(f6247,keyb,pevent,pecho)
 1000 call pwait(30.0, f6247, icl, idnr)
      if (icl .eq. pnclas) goto 2000
C
C then get string inetractively from the keyboard.
C
      call pgtst(lostr,str)
C
      chars(1:lostr) = str(1:lostr)
      call patr3(0.0,0.0,0.0,-0.1,0.1,0.22,chars)
      goto 1000
 2000 continue
      call pdel
      call pclst
C
C epilogue
C
      call pclwk(f6247)
      call pclph
      stop
      end
```

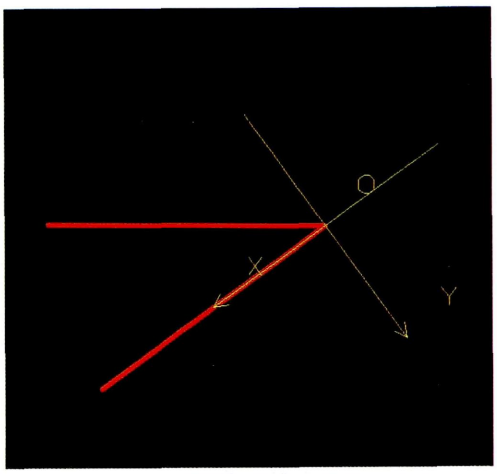
C　メタファイルを GKS を通して出力する

以下のプログラムはメタファイルを読み込んで各種ワークステーションに出力する GKS のプログラムである。GKS では PHIGS と異なり、ワークステーションとして VDS やプリンタが用意されている。このプログラムはワークステーションを対話的に選択できるようになっている。

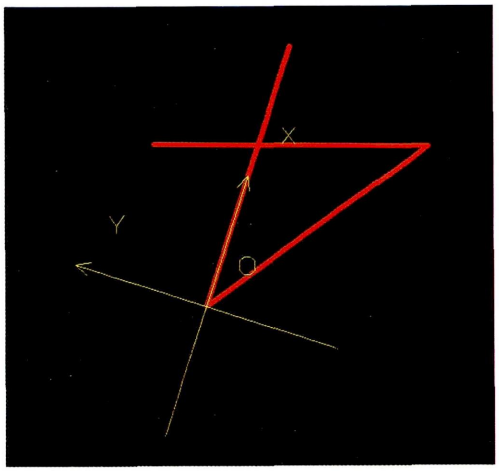
```
      parameter(lendim=80)
      integer wkid,itype,ldr,stat
      integer conid,wtype,wkid2,conimf,mi
      character*8 char
      real window(4),viewpt(4)
      data wkid,conid,wtype/1,1,1/
      data wkid2,conimf,mi/2,2,2
      data window/0.0,30.0,0.0,30.0/
      data viewpt/0.0,1.0,0.0,1.0/
      character datrec*80(lendim)
C
      call gopks(0,1)
      call gopwk(wkid2,conimf,mi)
      call gopwk(wkid,conid,wtype)
      call gacwk(wkid2)
      call gacwk(wkid)
      call gswn(1,window(1),window(2),
     -            window(3),window(4))
      call gsvp(1,viewpt(1),viewpt(2),
     -            viewpt(3),viewpt(4))
      call gselnt(1)
 1000 continue
      call ggtitm(wkid2,itype,ldr)
      if (itype .eq. 0) goto 9000
      if (itype .eq. 100) then
         call grditm(wkid2,0,lendim,datrec)
      else if (ldr .gt. lendim*80) then
         call grditm(wkid2,0,lendim,datrec)
      else
         call grditm(wkid2,ldr,lendim,datrec)
         call giitm(itype,ldr,lendim,datrec)
      endif
      goto 1000
 9000 continue
      call grqch(1,1,stat,char)
      call gdawk(wkid2)
      call gdawk(wkid)
      call gclwk(wkid2)
      call gclwk(wkid)
      call gclks
      stop
      end
```



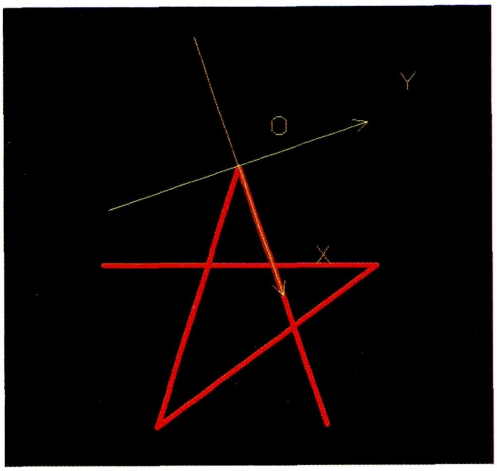
図A-1



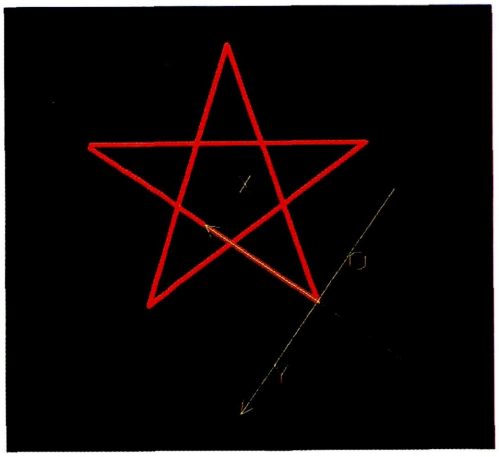
図A-2



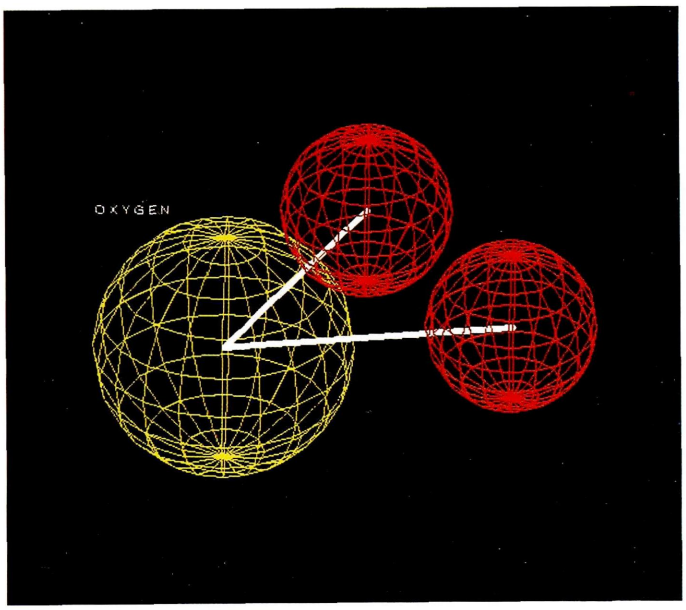
図A-3



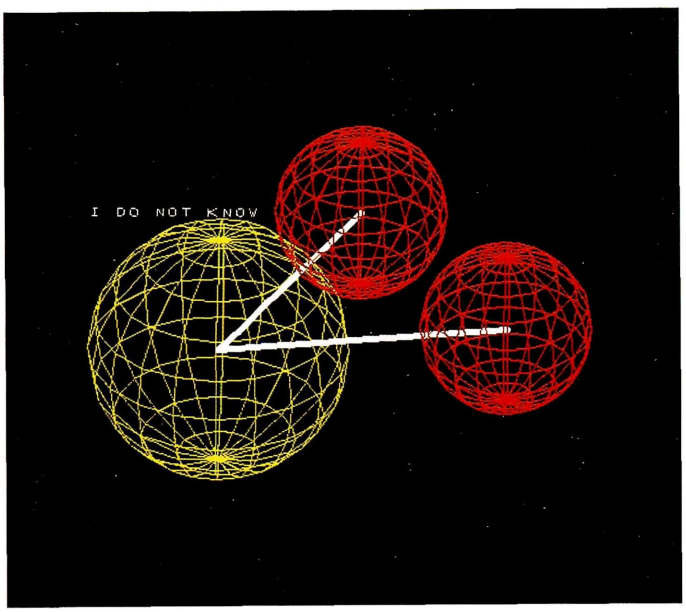
図A-4



図A-5



図B-1



図B-2