Geometry-Oriented Graphics Interface

鎌田 富久　　川合 慧
（東京大学 理学部）

1．はじめに

何らかの形で図形を表示する応用プログラムはグラフィクス・パッケージ（システム）と結合される。グラフィクスの国際標準としてGKS［1，2］があり、さらにGKS－3D［3］、PHIGS［4］などの標準化案が現在検討されている。これらの標準はワークステーション（デバイス）の側から設計されており、必ずしも応用プログラムから利用しやすいわけではない。図形出力に関して言えば、これらの標準は出力プリミティブと属性設定のためのコマンド列から成っており、応用プログラム側では図形を描く順番を意識しなければならない。PHIGSでは階層構造（structure)が取り入れられているが、この機能はコマンド列のサブルーチン呼び出しにすぎず、図形を記述する上では非常に低レベルである。

我々はグラフィクス・システムを応用プログラムの側から検討し、［5］に見られるような階層構造を発展させた高レベルなインターフェイスを提供するグラフィクス・システムを考案した。本稿では我々が現在研究開発している3次元グラフィクス・システムGRIP II(Graphics for Intelligible Picture)について述べる。
（GRIPシステムについては［9，10］）

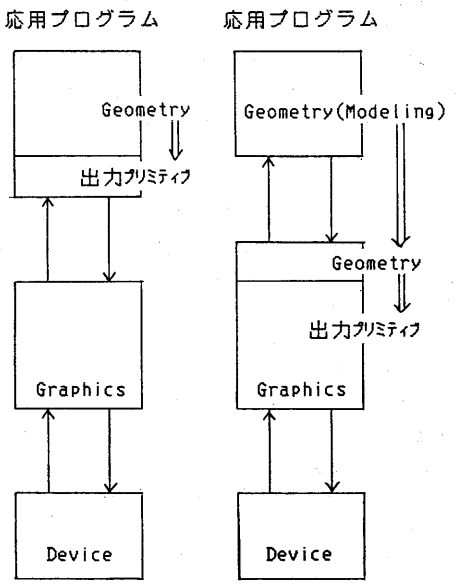


図1．GRIPIIのインターフェイス

2．GRIP II の特徴

3次元の図形を表示する場合、応用プログラムは3次元図形のモデリングを行っている。従来のグラフィクス・システムでは、応用プログラム側で幾何データを出力プリミティブなどのコマンド（関数）列に翻訳しなければならない。この方式ではグラフィクス・システム側で隠線・隠面処理は扱いにくく、単純な隠線・隠面消去しかできない。

GRIPIIでは応用プログラムの幾何データをGeometryのまま取り込むインターフェイスになっている（図1）。このようにGeometry主体のインターフェイスをとれば、Geometry内部の関係、あるいはGeometryどうしの関係を考慮した隠線・隠面処理、陰影付けが可能になる。現在GRIPIIのGeometryの表現はサーフェス・モデルを採用している。応用プログラムでソリッド・モデリングなどのもっと複雑なモデリングを行っている場合には、出力図形の境界表現からGRIPIIのGeometry表現に変換する必要がある。

GRIPIIの第2の特徴は図形の階層的記述である。それは次の2つの側面を持つ。

・幾何データの階層的記述
・属性データの階層的記述

GRIPIIでは幾何データと属性データを分離して記述する立場をとっている。幾何データは幾何変換を通して階層的に定義されるので、図形の個々の局所的な部分について、独自の座標系で図形を記述することができる。また、属性データは個々の線、あるいは面に対して1つの値が指定されるのではなく、それが隠れている情報に依存して線、あるいは面を分割して別々の属性の値を指定することが可能な機構になっている。しかも、このような属性の指定が階層的にできる点が大きな特徴である。

３．ＧＲＩＰ II の階層構造

ＧＲＩPIIにおける図形の階層的記述は worldとgeometryによって表現される。 worldとは１つの中間的あるいは最終的座標系であり、他の worldまたはgeometryを変換することによって構成される。geometryとは論理的にまとまった幾何データの集合で、独自の座標系をもっている。したがって、 worldとgeometryによる図形の記述は図２のようにノードに world、リーフにgeometryという形式の階層になる。制限としてサイクリックな構成は禁止している。表１はgeometryと worldの形式的定義である。

３．１　geometryの構造

geometryは図形の基本的要素を表現する幾何データの集合で、現在サーフェス・モデルを採用している。つまり、点、線、面のデータを次のように定義する。

- <point>　p = (x,y,z)
- <line>　l = (p1,p2)
- <surface> s = (l1,l2,...,ln)

点の座標は３次元座標とし、２次元の図形を特別扱いしない。また面はいくつかの穴を持つことが許されており、面の内外の判定は、半直線と境界線分の交点の数の偶奇によるものとする。geometryには名前が付けられ、 worldから参照される。

３．２　worldの構造

worldはいくつかの構成要素(world element) から成り、この構成要素は次の４つの部分から成る。

- <element name>　要素名
- <child type>　下位階層のタイプ
- <child name>　下位階層の名前
- <transformation>　変換データ

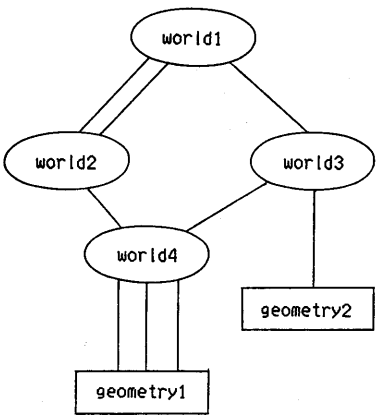


図２．　worldとgeometryによる階層構造

```
<geometry>::=<geometry name>
                <points><lines><surfaces>
<points>::=<point>*
<lines>::=<line>*
<surfaces>::=<surface>*
<world>::=<world name><world element>*
<world element>::=<element name><child type>
                <child name><transformation>
<child type>::= world | geometry
<transformation>::=<geometric transformation>
                <picturing transformation>
```

表１．world と geometry の定義（BNF）

<element name>はその worldを参照する上位階層で属性の記述をするのに用いられる。<child type>は下位階層がgeometryか worldかを示し、その名前が<child name>である。最後の<transformation>が worldの実体を表現しているもので、それは次の２つの変換から成る。

- <geometric transformation(GT)>
  幾何変換
- <picturing transformation(PT)>
  図化変換

<GT>は図形を下位階層の座標系からその worldの座標系に変換するためのパラメータ・データである。<PT>は下位階層の図形に対して属性を指定するためのデータで下位階層の<element name>が参照される。<GT>と<PT>は独立ではなく、<GT>を実行した後の奥行き（Ｚ座標）情報によって<PT>でViewに依存した指定ができるようになっている。これら２つの変換については３、４で詳しく述べる。 worldには名前が付けられ、他の worldから参照される。

３．１でgeometryの定義を行ったが、最下位の階層であるgeometryの幾何データの集合の大きさがどの程度であるかは規定されていない。そこで、最下位階層が個々の面（または面に含まれない独立な線あるいは点）になるような階層構造を考える。

例えば、立方体を４つの側面と上下面で属性の指定で別の扱いをしたい場合には、図３のような階層化を考える必要がある。この場合の正方形のように厳密な階層化の最下位階層をprimitive geometryと呼ぶ。この場合のworld cubeは<world element> として２つの要素を持つことになる。

このような厳密な階層化は実用的でないので、ＧＲＩPIIでは下位階層をgeometryとしてまとめて定

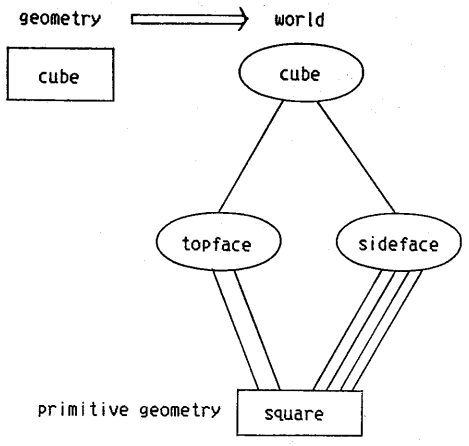


図３．立方体の厳密な階層化

義する方法を提供している。そこで、厳密な階層化を行った場合の<world element> にかわるものとしてgeometryにも<geometry element>を導入する。

<geometry element>::=<element name>
<points><lines><surfaces>

<world element> と同じように<element name>は上位階層での<PT>で参照される。

３．３　図形の出力と修正

ＧＲＩPIIで図形の変換をするには<world name>を指定すればよい。その場合指定されたworld 以下の階層で表現される図形を出力する。つまり、全階層の根以外のworld を指定することができる。また、出力に指定されたworld の座標系が出力表示装置の座標系になり、一般にＺ座標は無視されて出力される。

次に worldの図形出力プロセスについて説明する。ＧＲＩPIIではGeometryとGeometryの隠線・隠面情報（shielding information)によって図形の属性を決定する仕組みになっているので、図形を出力するのに階層構造をトラバースしながら順々に図形を描くわけにはいかない。図形出力プロセスは階層の最下位の worldから順に実際にその worldを構成することによってなされる。 worldの構成は幾何データを実際にその座標系に変換し、指定された属性を決定することである。図４は図２で示される階層構造のworld1を出力するときのプロセスを表している。

図形の修正については次の３つの場合がある。

（１）geometryの修正
（２）world element の transformation の修正
（３）world element の追加、削除

（１）、（２）は階層構造の内容の修正で、（３）は構造自体の修正である。修正を加えた図形の再出力のプロセスは、修正された worldと修正された worldまたはgeometryを下位階層にもつ worldを再構成することによって得られる。たとえば図４の例で、geometry2(G2) を修正すれば、world3(W3)とworld1(W1)が再構成され、world4(W4)とworld2(W2)は既に構成されているものがもちいられる。

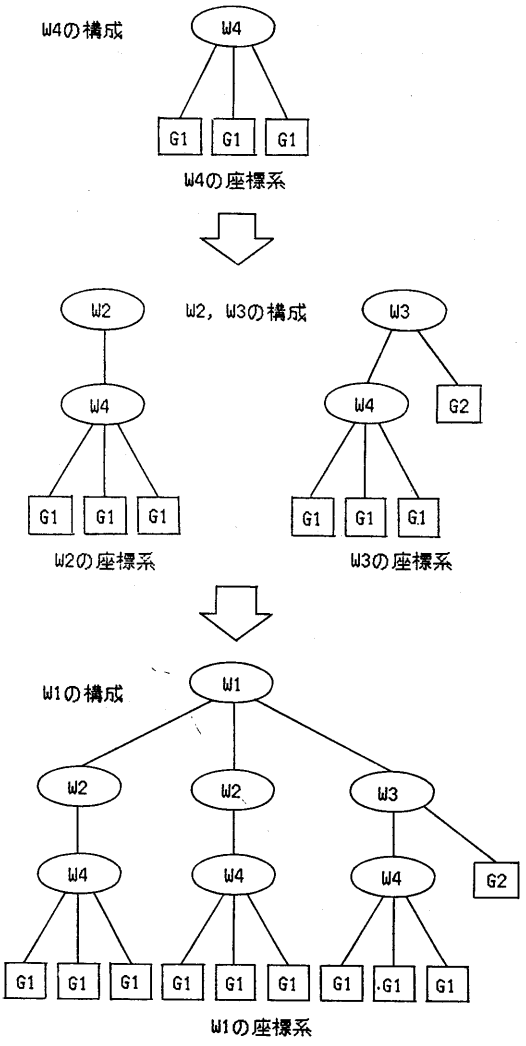


図４．world1(図２)の出力プロセス

４．Geometric Transformation

下位階層の幾何データに幾何変換を施して world の座標系に写像するのがgeometric transformation である。ＧＲＩＰIIでは、幾何変換は４×４マトリクスによる同次座標変換(homogeneous coordinate transformation) で表現される変換を扱う。同次座標変換については［６，７，８］。同次座標変換では拡大・縮小(scaling) 、回転(rotation)、せん断(shearing)、反転(reflection)、平行移動(translation) 、透視(perspective trans.)が扱える。伝統的にグラフィクスの座標変換に用いられているウインドウ・ビューポート変換は、拡大・縮小と平行移動の組み合わせで同次座標変換の非常に狭いサブセットである。さらに、我々はビューイング変換(viewing trans.)を装置座標系に変換するときだけに行うという考えを捨て、中間座標系への変換にも自由にビューイング変換を用いることによって多彩な図形記述を可能にした。また、同次座標変換はかなり自由度のある変換であるから、変換を指定する仕方には柔軟性が要求される。ＧＲＩＰIIでは、次に述べるようにいくつかの変換を表現する方法が取り入れられている。

４．１　変換の種類

変換は４×４マトリクスによって表現されるものであるから、次に示すのはマトリクスを記述するためのユーザ・インターフェイスである。

（１）ウインドウ・ビューポート変換（図５(a)）

下位階層の座標系の軸に平行な直方体（Xmin,Xmax,Ymin,Ymax,Zmin,Zmax)を worldの座標系の軸に平行な直方体（Xmin',Xmax',Ymin',Ymax',Zmin',Zmax')に写像する。Xmin'=Xmax' またはYmin'=Ymax' またはZmin'=Zmax' のときは２次元平面への写像となる。

（２）基本変換の組み合わせ（図５(b)）

基本変換として拡大・縮小、回転、反転、平行移動を用意し、これらの変換の組み合わせ順序を指定する。多くの場合、幾何変換はこの変換で十分表現できる。

（３）軸方向ベクトルの変換（図５(c)）

下位階層の座標系の軸方向単位ベクトルex, ey, ez が写像されるべき world座標系でのベクトルex', ey', ez'と、原点が写像されるベクトルdを指定する。この場合、ex', ey', ez'の直交性は要求されない。

（４）ビューイング変換（図５(d)）

ビューイングのパラメータとして視点の位置、ビューボリュームの大きさ、ビューイングタイプ（parallelまたはperspective)などを指定する。ビューボリュームが worldの座標系の軸に平行な直方体(Xmin',Xmax',Ymin',Ymax',Zmin',Zmax')に写像される。ビューボリュームの奥行き方向がビューポートのＺ軸方向に写像され、Zmin'=Zmax' のときいわゆる射影変換になる。

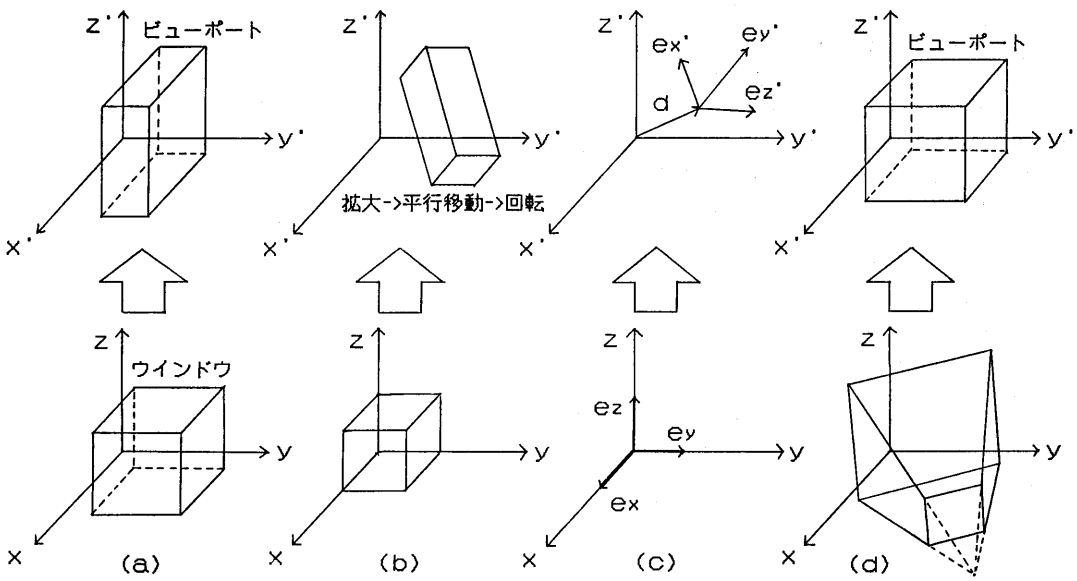


図５．Geometric Transformation の種類

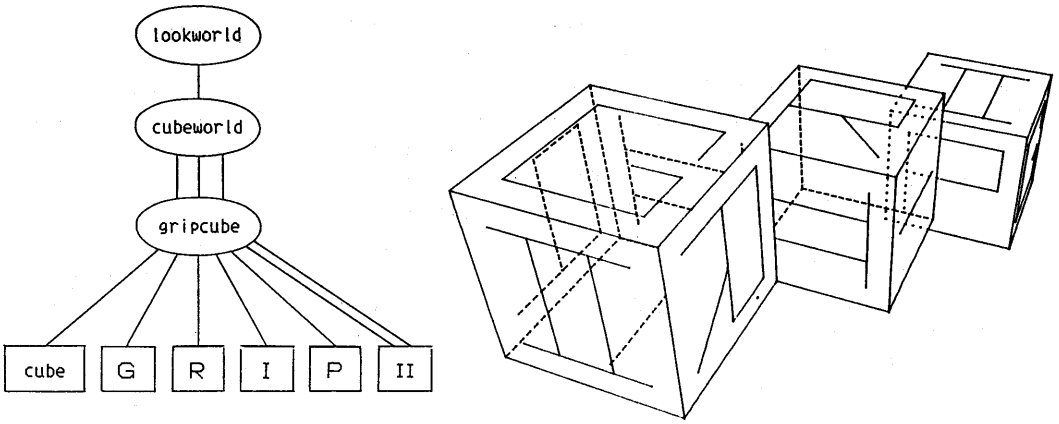


図6．立方体を基本にした簡単な図形

（５）４×４マトリクスの指定
４×４のマトリクスを直接指定する。

（１）は（２）のサブセットであり、（２）は（３）のサブセットである。（２）は（１）に回転、反転が加わったもので、（３）は（２）にせん断が加わったものである。（４）に関しては奥行き情報によって属性を指定する場合に有用である。

４．２　例

図６に立方体を基本にした簡単な図形の例を示す。geometryとして立方体cubeと、xy平面上の線の集合であるG、R、I、P、IIがある。 worldであるgripcubeは７つのworld element をもち、cubeに関しては変換（１）、G、R、I、P、IIに関しては変換（２）を用い、立方体の各面に写像する。cubeworld はchild がgripcubeである３つのworld element をもち、変換（２）を用いて立方体を空間に配置する。最後にlookworld はcubeworld を変換（４）によって透視変換を行って写像する。

ＧＲＩＰIIでは、階層的な幾何データの記述によってモデリングの機能をある程度実現している。そして、さらに広範囲のアプリケーションを扱うためにはgeometryの表現を柔軟にしていくことが必要である。

５．Picturing　Transformation

図形を構成する基本要素である点、線、面、に対してグラフィカルな属性を指定する機構について述べる。

３次元の図形を２次元の表示装置で表示する場合、失われる奥行きの情報をどう表現するかが問題である。奥行きを計量的に表示できない以上、それは属性で補うしかない。つまり、奥行き、隠れ情報を属性の値にいかに反映させるかが鍵になる。それは従来のグラフィクス・システムが行っているような画一的な隠線・隠面消去では不十分であり、もっと柔軟な扱いが必要である。

５．１　概説

個々の点、線、面に１つの属性の値を指定するだけでなく、隠点、隠線、隠面情報を考慮して属性の指定を行う。奥行きはgeometric transformationが実行された後のＺ軸方向の値で計算される。

線、面に関しては面に隠れている情報によって線、面を分割して別の属性を指定できる。図７(a)は線分ABが面s1，s2に隠れているかどうかによって５つの部分に分割される様子を示し、図７(b)は面s1が面s2，s3に隠れているかどうかによって３つの部分に分割される様子を示している。これらの分割されたそれぞれの部分に１つの属性値を指定する。

このようにＧＲＩＰIIの属性の指定は図形の分割を巻き込む一種の変換である。我々はこれをpicturing transformation(PT)と呼ぶことにする。PTは奥行き、隠れ情報から属性という視覚化要素への写像であり、非常に強力で柔軟な図化機構を提供するＧＲＩＰIIの最大の特徴である。

５．２　ＰＴの記述システム

点、線、面の属性として次のものがある。

- point --- point_type point_color point_size
- line --- line_type line_color line_width
- surface --- surface_type surface_color surface_index

point_type, line_type, surface_type には属性値としてtransparent があり、表示したとき見えないことを意味する。point_typeとは通常のマーカータイプのことである。上の9つの属性は別々に指定される。

まず、奥行き情報を考慮しないで指定する場合について述べる。その形式は次のような式(PT statement)で構成される。

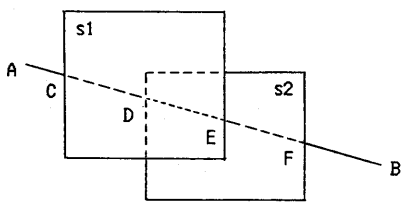
```
[要素名　要素名　．．．] ＝属性値
[要素名] ＝属性値
ALL　＝属性値
```

要素名(element name)は下位階層のworld element またはgeometry name で、下位階層のすべての要素をパス名式で参照できる。ALL は下位階層のすべての要素を表す。また式の意味は、点の属性のときは指定された要素に含まれる点の属性値を指定された値にすることを示し、線、面についても同様の意味になる。

PTは各 worldで実行されるので、ある点、線、面はすでに属性値を指定されているかもしれない。その場合には上の式は無視される。つまり、下位階層のPTが優先される。上位階層のPTで再指定したい場合には次の式を用いる。

```
[要素名　要素名　．．．] ＜属性値
[要素名] ＜属性値
ALL　＜属性値
```

(a) 線の場合



(b) 面の場合

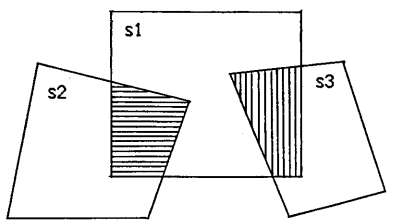


図7．PTによる図形の分割

```
<PT description>::=<PT statement>*
<PT statement>::=
              <condition>＝<attribute value>
             |<condition>＜<attribute value>
<condition>::=<element condition>
   |<element condition>－<shielding condition>
<element condition>::= ALL | [ <elements> ]
<elements>::=<element name>
             |<element name><elements>
<shielding condition>::=<tag> [ <elements'> ]
<tag>::= ANY | ICD | SET
<elements'>::=<element name> | ？
             |<element name><elements'>
```

表2．PTを記述するための形式の定義（BNF）

この式はすでに属性値をもっているものにも適用される。さらに、すでに指定されている属性値をも考慮に入れた再指定の機能を現在検討している。

次に隠れ情報を考慮した指定の形式について述べる。それは上に述べた形式に何に隠れているかを表す部分を付け加えることによって得られる。それは次のような形式によって表現される。

```
[要素名　．．．]－tag[要素名　．．．]＝属性値
[要素名] －tag[要素名　．．．]＝属性値
ALL　－tag[要素名　．．．]＝属性値
```

（ ＝ のかわりに ＜ としてもよい。）

－tag 以下の部分が隠れ条件を表していて、意味はtag の種類（ ANY, ICD, SET）によって次のようになる。

- ANY --- 以下の要素のうちの少なくとも1つに隠れる。
- ICD --- 以下の要素のすべてに隠れる。
- SET --- 以下の要素のすべてに隠れ、かつそれ以外の要素に隠れない。

条件の強さは SET, ICD, ANYの順となる。上で用いた要素に隠れるというのは、要素に含まれる面のどれかに隠れるということである。また、tag 以下の要素名に特別な記号として？を用いることができる。これは不特定の要素（面を含む）を表す。したがって、すべての隠線を除きそれ以外を実線で表示するためのPTの指定は、

```
<line_type>
  ALL － ICD[ ？ ] ＝ transparent
  ALL ＝ solid
```

world name	element name	child type	child name	PT
lookworld	look	world	cubeworld	→
cubeworld	cube1	world	gripcube	
	cube2	world	gripcube	
	cube3	world	gripcube	
gripcube	body	geometry	cube	
	letterG	geometry	G	—
	letterR	geometry	R	—
	letterI	geometry	I	—
	letterP	geometry	P	—
	letterII1	geometry	II	—
	letterII2	geometry	II	—

また、geometry cube には element として plane がある。

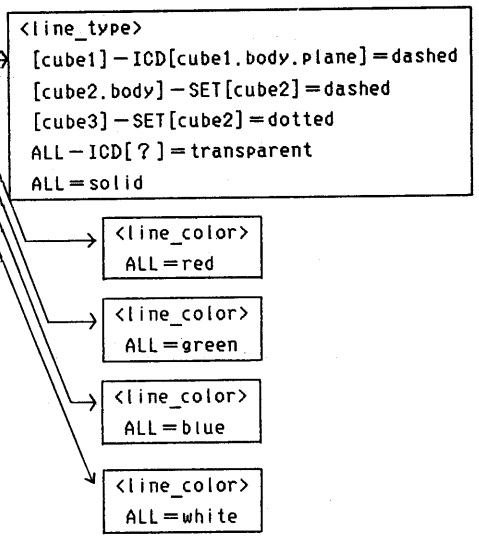


表３．　図６の図形の worldの構成とPT

となる。この例からもわかるように、PT statement は最初の式から順に評価される。

５．３　形式的な定義

ここでは５．２で説明したPTを記述するための形式を形式的に定義する。表２にBNF による定義を示す。

次に１つのPT statementが成り立つ条件を形式的に定義する。

(<)

[a1 a2 ... an] － tag[b1 b2 ... bn] ＝ value

－－－－ (＊)

a1,a2,...,an : element name

b1,b2,...,bn : element name

まず、a をelement nameとするときにP(a)、L(a)、S(a)はそれぞれa に含まれる点の集合、線の集合、面の集合を表すものとする。例として線の属性の場合を取り上げると、線l のある区間が面の集合v にちょうど隠れているとき（どの面にも隠れていないとき vは空集合）、式(＊)が成り立つのは次の条件が成り立つ場合である。

(element condition)∧(shielding condition)

(element condition)≡ l∈(L(a1)∪...∪L(an))

(shielding condition)≡

tagがANYのとき v∩(S(b1)∪...∪S(bn))≠φ

ICDのとき v∩S(b1)≠φ∧...∧v∩S(bn)≠φ

SETのとき v∩S(b1)≠φ∧...∧v∩S(bn)≠φ

∧v⊂(S(b1)∪...∪S(bn))

<element condition> が ALLのときには条件(element condition) は真であり、<shielding condition>がない場合は、条件(shielding condition) は真である。同様に点、面の場合には、(element condition) の定義で Lのかわりにそれぞれ P、S を用いればよい。

５．４　例

図６に示した図形のPTについて説明する。各 worldの構成は表３のようになっている。属性としては line_type とline_colorを考える。

line_type に関しては最上位階層のlookworld で隠線情報を考慮した指定をしている。要素のパス名cube1.body.planeは要素cube1 の要素bodyの要素plane という意味で、plane はgeometry cube の要素(geometry element)で立方体の２つの面（Gと１つのIIに対応する）からなる。つまり、第１式はcube1 の線でcube1 の２つの面（GとII）に隠れている部分を破線にする。第２式は、cube2 のbody（立方体のわく）でcube2 だけに隠れている部分を破線にする。第３式は、cube3 の線でcube2 だけに隠れている部分を点線にする。第４式はそれ以外の隠線を表示せず、第５式は隠れていない線を実線にする。

line_colorに関しては、gripcubeのbodyで白色の指定があるので、これはcubeworld での色の指定より優先される。つまり、立方体のわくはすべて白色で表示され、文字を表す線についてのみcube1 では赤色、cube2 では緑色、cube3 では青色で表示される。

一般に個々の図形要素について属性の値を変えたい場合には下位の worldで属性を指定し、まとまっ

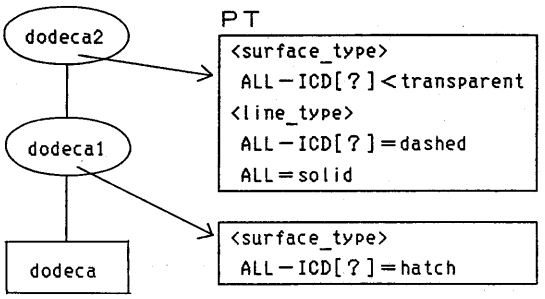


図９．陰付き正１２面体表示の階層構造とPT

た図形に対して属性を決めたい場合には上位の world で属性を指定する。

６．応用例（陰付け）

ＧＲＩＰIIでは属性を隠線・隠面情報によって決定した後で、さらに変換を通して別の worldに写像することができるので、陰付けした表示が可能である。つまり、最初に光の方向からみたビューイング変換を行って陰を付け、つぎに視点からみたビューイング変換を行って図形を表示する。

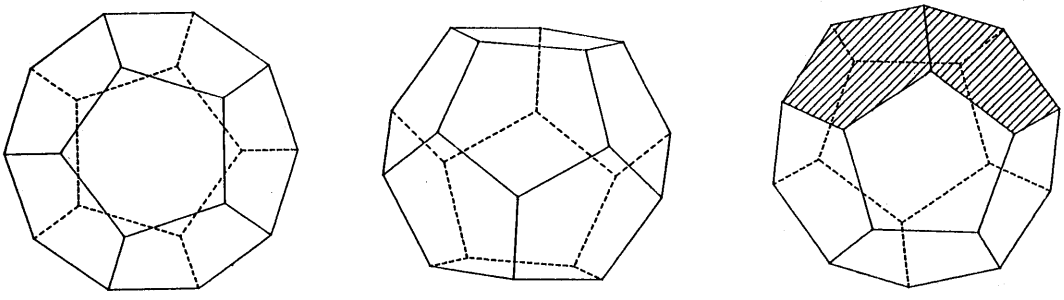
正１２面体（１２の面、３０の線、２０の点）を geometry dodeca とする。dodecaの座標系でＺ軸の方向から見た図が図８(ａ)である。次にこれをある方向（光の方向）から見た図にビューイング変換(parallel)して world dodeca1に写像する。このときPTで隠面に対して属性surface_typeにハッチを指定する。dodeca1 の座標系でＺ軸の方向から見た図が図８(ｂ)である。さらにこれをＹ軸の方向（視線方向）から見た図にビューイング変換(parallel)して world dodeca2 写像する。このときPTで隠面を表示しない指定と隠線を破線で表示する指定を行う。出力図形は図８(ｃ)のようになる。また、PTは図９のようにすればよい。したがって最終的にハッチングされる面は、dodeca1 では隠れていてdodeca2 では隠れていない面ということになる。

７．おわりに

ＧＲＩＰIIの階層構造と変換の組み合わせは出力図形を記述するうえで非常に強力なものである。特にPicturing TransformationはＧＲＩＰII独特の機能であり、これによってViewに関する様々な問題、隠線・隠面の凝った扱い、陰影問題などを処理することができる。

我々は応用プログラムの側から望まれる高レベルなグラフィクス・インターフェイスを目標に研究を進めており、その意味ではＧＲＩＰIIはまだ中間段階にすぎず今後さらに研究を深めていかなければならない。

ＧＲＩＰIIは表示デバイスが普通もっているような（階層）セグメント機能をはるかに超える機能をもっており、そのため現在ではグラフィクス・ワークステーションの機能としてＧＲＩＰIIの階層構造およびPTを実現することは難しい。しかし、昨今のグラフィクス・ワークステーションのめざましい発展を見れば、今後機能は加速度的に向上していくにちがいない。そうした高機能化の１つの方向をＧＲＩＰIIは指針しているといえる。



（ａ）geometry dodeca　（ｂ）world dodeca1　（ｃ）world dodeca2（最終図）

図８．正１２面体の陰付け表示

```
open_geometry(name)
close_geometry()
open_world(name)
close_world()
define_point(n,point)
define_line(n,line)
define_surface(n,surface)
define_geometry_element(name,
          points,lines,surfaces)
define_world_element(name,
          childtype,childname,transformation)
display(name)
```

付録．ＧＲＩＰIIの主なコマンド

参考文献

[1] Information Processing - Graphical Kernel System(GKS) ISO DIS 7942, 1982

[2] G.Enderle etal, Computer Graphics Programming. Springer-Verlag, 1984

[3] GKS-3D ISO/TC97/SC21/WG5-2 N277 Rev., 1985

[4] PHIGS ANSI X3H3/83 1983

[5] P.T.Hagen etal, ILP Intermediate Language for Pictures. Mathematical Center Tracts 130, Amsterdam, 1980

[6] J.D.Foley and Van Dam, Fundamentals of Interactive Computer Graphics. Addison-Wesley, 1982

[7] W.M.Newman and R.F.Sproull, Principles of Interactive Computer Graphics. McGraw-Hill, 1979

[8] D.F.Rogers and J.A.Adams, Mathematical Elements for Computer Graphics. McGraw-Hill, 1976

[9] 鎌田 川合, Grahics for View Dependent Aspect Determination. 情報処理学会第29回全国大会 pp.1621-1622

[10] 鎌田 川合, Grahics for View Dependent Aspect Determination－ＧＲＩＰシステムの拡張 情報処理学会第30回全国大会 pp.2043-2044