

共通中間言語を利用した3次元画像記述の一手法と 高速画像生成装置MAGICへの適用

秋本 高明 玉邑 嘉章 末永康仁

NTT 電気通信研究所

コンピュータ・グラフィクスにおいて、3次元画像の記述は重要な問題である。

本文では、LIGHT(a Language as an Interactive Graphics Handling Tool)と呼ぶ新しい3次元画像記述用中間言語を提案する。このLIGHTを直接使用することにより、各種の3次元画像を容易に記述することができる。さらに、LIGHTをC・FORTRAN・PASCALなど任意の手続き型言語の中で使用することにより、動きを持ったより複雑な3次元画像を記述することもできる。

LIGHTは、次に示すような3次元コンピュータ・グラフィクスに必要なデータをコンパクトに記述する。(1)物体の形状、(2)色・反射率などの属性、(3)光源に関するデータ、(4)視点位置などの観測条件。

3次元画像記述言語LIGHTは高速画像生成装置MAGICを使用した画像生成実験において有効に使用されている。

3D Graphics Data Description
by A New Intermediate Language (LIGHT),
and Its Application
to A High Speed Graphics Computer MAGIC

Taka-aki AKINOTO, Yoshiaki TAMANURA and Yasuhito SUENAGA
NTT Electrical Communications Laboratories (Yokosuka-shi, 238-03, Japan)

3D graphics data description is one of the most important problems in interactive computer graphics.

This paper presents a new intermediate graphics language called a LIGHT (a Language as an Interactive Graphics Handling Tool). Various 3D graphics can be easily described by using LIGHT directly. LIGHT can be also used in arbitrary procedural languages such as C, FORTRAN, PASCAL, in order to describe more complicated 3D graphics having motion.

The following elements needed for interactive computer graphics are compactly described in LIGHT. (1)Shape of objects. (2)Attributes (color, reflectance, etc). (3)Light source parameters. (4)Camera parameters.

LIGHT is successfully applied to 3D graphics generation experiment using a high speed graphics computer MAGIC.

1. まえがき

コンピュータなどの演算装置及び画像出力装置の高性能化・低価格化に伴い、立体的な画像を人工的に生成する技術が盛んに使われるようになった。更に、高速画像生成アルゴリズム、画像生成専用ハードウェアおよび各種の材質の表示技術等の研究により、いろいろな物体のリアルな画像が高速に生成できるようになりつつある。

しかし、画像生成が容易になるにつれて、画像のもとになるデータの作成の手間が大きな問題となってきた。つまり、物体の3次元形状、その表面の色や反射率、物体の位置、視点位置や注視方向などの観測条件、およびそれらの時間的変化等のデータ（以後3次元画像データと呼ぶ）の作成にはいまだに多くの時間と労力が必要である。

3次元画像データ入力方法としては、ラフスケッチから物体の3次元形状を復元しそれに色・反射率などの属性を付加する方法や、カメラや3次元情報入力装置等から得た画像またはデータから物体の3次元形状とその表面属性を復元・抽出する方法等が理想的であるが、現在の技術ではまだ難しい。そこで、CAD等の分野では、操作性の優れた会話型の形状入力方法の開発が進んでいる。

もう一つの入力方法として、セルラ・アレイ・プロセスCAPで使われているMEG⁽¹⁾やLINKS-1で使われているImage Score⁽²⁾のように、人間に分かりやすい形でシーンのようすを記述できる言語（以後3次元画像記述言語と呼ぶ）を定め3次元画像データをその言語で記述する方法がある。MEGは、パラメトリック・アニメーション生成機能を持つ、オブジェクト指向の考え方により階層的に物体を定義できる、物体の各部分の位置をシンボルで指定できる、などの特徴を持つ言語で、C言語を拡張することにより実現している。Image Scoreは、MEGと同様に物体を階層的に定義する、テクスチャや属性を非常に細かく操作できる、などの特徴を持つ。両言語とも非常に完成された記述言語である。

このような方法によるデータの入力の容易さは会話型より劣るであろうが、会話型の形状入力には専用のグラフィックディスプレイやポインティングデバイス等が必要なものに対して、言語で記述する方法はそれを解釈して画像生成に必要な具体的な数値データ（以後画像生成用データと呼ぶ）に変換するプログラムがあればよい。つまり、容易に記述できる言語を定めれば、特別なハードウェアを必要としない手軽でポータ

ビリティの優れた方法といえる。

さらに、3次元画像記述言語によるシーンの定義は単なる文字列であり、一般の計算機プログラムでこれを自動的に作り出すことは簡単である。通常のプログラムで、ある言語によって記述された3次元画像の定義を作り出す方法は、

(1) 共通の中間言語を定めそれを発生する各種のプログラミング言語用副プログラムを用意することにより、C、FORTRAN、PASCALなど各種のプログラミング言語で画像生成用データを作り出すことができる、

(2) 中間言語の記述能力に加えてプログラミング言語の計算機能や実行制御機能などを利用することができる、

などの利点がある。

本文では、3次元画像記述方法の一つとして、3次元画像記述用の共通中間言語LIGHT(a Language as an Interactive Graphics Handling Tool)を定め、通常のプログラムによりこの言語で記述された3次元画像記述を作成し、それを解釈して画像生成用データを作成する方法を提案する。さらに、高速画像生成装置MAGIC⁽³⁾⁽⁴⁾へのインプリメントについて述べる。

2. 共通中間言語LIGHTを利用したデータ作成方法

画像生成には次に示すデータが必要である。

- (1) シーン中に存在している物体の3次元形状とその表面の色・反射率・屈折率・内部の色などの属性。
- (2) 各物体の位置。
- (3) 光源の位置または方向と強度。
- (4) 視点の位置・注視点の位置・上方向・ズームなどの観測条件。

さらに、動画を作成する場合、上記(1)～(4)のデータを逐次変化させることが必要である。

そして前述したように、上記のデータの入力にはいまだに多くの労力と時間がかかっており、それを解決する一つの方法として3次元画像記述言語によりシーンの様子を定義する方法がある。

ある言語で上記のデータを定義する場合、その言語は次のような性質を持つことが望まれる。

(1) 簡潔かつ容易に記述でき、後から読みなおしたときでも分かりやすい。

(2) 既に定義した形状を組み合わせてより複雑な形状を定義するといった階層的な定義ができると共に、

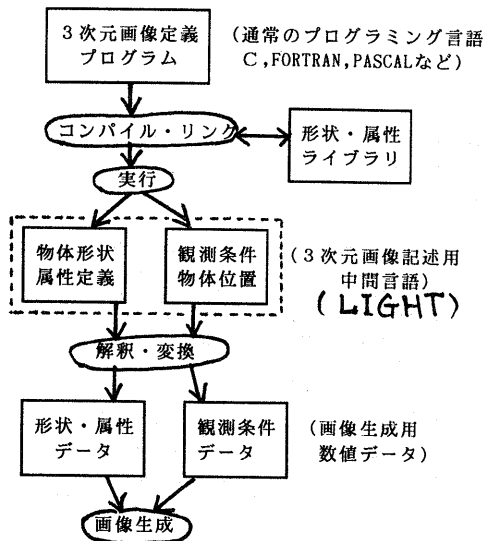


図1 中間言語を用いた
3次元画像データ入力方法

属性の継承機能を持つ。

(3) 定義された形状・属性を「部品」として蓄積しておき、後からそれらを容易に組み込むことができる

(4) 寸法、位置、色、反射率などを可変とした汎用的な部品を定義できる。

(5) 繰り返しによる類似した複数の部品の定義や、条件付き定義ができる。

(6) 動画を作成する場合、時刻による物体位置や視点位置などの変化が容易に記述できる。

しかし、これらの条件をすべて満足する3次元画像記述言語とその処理系は複雑で大きなものとなる。一方、既存の計算機言語は、引き数による副プログラムへのパラメータの受渡し、繰り返し・条件分岐などの実行制御の機能を備えており、この機能により上記(4)～(6)の条件が満足できる。

つまり、上記(4)～(6)の機能はないが3次元画像を容易に定義できるコンパクトな言語を定め、これとプログラミング言語を組み合わせれば上記の各条件を満足できる。即ち、既存のプログラミング言語で物体形状、位置、観測条件およびそれらの時刻に沿った変化を記述し、変数や計算式で記述された各種の値や繰り返しまたは条件付き定義をプログラムを実行することで計算または展開して3次元画像記述用中間言語による具体的な記述に変換し、それをさらに解釈して画像生成用データを作成する方法である。この方法による3次元画像データ入力の手順を図1に示す。

3. 3次元画像記述中間言語LIGHT

前述した入力方法で使用する3次元画像記述用中間言語LIGHTを次のような方針で設計した。

(1) 直接この言語で人間が3次元画像を記述できるように、シンボリックに分かりやすい表現で記述できるようにする。

(2) なるべく単純な処理で数値データに変換できるような書式にする。

(3) 1枚の静止画を具体的に記述できればよく、パラメータに変数を使用することや動画の記述は含めない。

(4) 小さな部品を組み合わせでより大きく複雑な物体を定義してゆくように、物体形状を階層的に定義する。

(5) 物体形状、属性等を部品化できるような記述方法とする。

さらに、これで作成された画像生成用データは主にレイトレーシング法で使われることと、このような言語型のモデリングに適しているという観点から、形状の表現形式はCSG表現を使用した。従って、形状の組合せは存在領域の集合演算により行う。また、階層構造に物体属性の継承機能を持たせた。つまり、もし下位の物体に属性が設定されてなければ上位の物体の属性が受け継がれる。上位の物体にも属性が設定されていない場合は、デフォルトの属性が下位の物体まで伝えられる。

表1に中間言語の記述規則を示す。このように、本3次元画像記述言語では、物体形状とその属性を定義する物体部分と光源・物体位置・視点位置などを定義する観測条件定義部分で情景を記述する。

物体定義部分は、(1)属性定義、(2)物体生成、(3)形状組合せ、(4)グループ化、(5)物体移動回転、(6)表示物体指定の6種類の文で記述される。表1で[]で囲まれた部分は省略ができる。

属性定義文は物体の色・反射率・透過率を輝度計算式の各係数の値で定義する。これらの係数は分りにくいので、いくつかの材質や色に対する属性を予め定義しておく必要に応じて参照するようにする。

物体生成文は、基本形状をもつ物体を生成しそれに必要ならば属性を付加する文である。基本形状は現在表2に示す8種類が用意されている。

形状組合せ文は、既に定義されている物体を集合演算によって組合せ新たな物体を生成し、必要ならばそ

表1 中間言語記述規則一覧

◆物体形状および属性の定義	
(1) 属性定義 attribute 属性名 [color r g b] [ia n] [kd n] end	(4) 物体等のグループ化 group 名前 物体・グループ名リスト [extent x y z r] end
(2) 物体生成 primitive 名前 プリミティブ名 パラメータ [attribute 属性名] end	(5) 物体・グループの移動回転 put 名前1 名前2 [tran x y z] [xrot n] [yrot n] [zrot n] end
(3) 形状組合せ construct 名前 集合演算式 [extent x y z r] [attribute 属性名] end	(6) 表示物体・グループ指定 display 物体・グループ名リスト
◆物体位置・観測条件定義	
(1) 光源設定 light タイプ x y z r g b	(5) ズーミング指定 zoom n
(2) 視点位置指定 from x y z	(6) 物体の移動回転指定 tran 物体名 x y z xrot 物体名 n yrot 物体名 n zrot 物体名 n
(3) 注視位置指定 to x y z	
(4) 上方向指定 up x y z	

表2 プリミティブ名とその形状

プリミティブ	パラメータ	形状 (存在領域)
plane	a b c d	$a*x + b*y + c*z + d \leq 0$
xyplane	なし	$z \leq 0$
yzplane	なし	$x \leq 0$
zxplane	なし	$y \leq 0$
sphere	r	$x^2 + y^2 + z^2 \leq r^2$
ellipsoid	a b c	$(x/a)^2 + (y/b)^2 + (z/c)^2 \leq 1$
cylinder	a b	$(x/a)^2 + (z/b)^2 \leq 1$
cone	a b	$(x/a)^2 + (z/b)^2 \leq y^2$

れにエクステントと属性を付加する文である。この文中の集合演算式は既に定義されている物体名を'+', '*', '-'の3種類の演算子で結んだものである。これらの演算子はそれぞれ左辺と右辺の和、積、左辺から右辺を除く演算を表す。演算子の評価順序は常に左から右とし、先に評価させたい部分は括弧で囲むことにした。

グループ文は、ひとかたまりとして扱ったほうが都合がよいいくつかの物体をまとめて、それに名前と必要ならばエクステントを付加する文である。グループをさらにまとめることもできる。

移動回転文は、物体またはグループをそのローカル座標系において移動・回転した場所にコピーする文である。表示物体指定文は、実際に表示させたい物体またはグループを指定する文である。

観測条件定義部は視点位置、注視点位置、上方向、撮影レンズの焦点距離に対応する係数、光源と物体位置を定義する。光源は複数個設定できる。

図2にこの言語で記述した3次元画像の記述例を示す。例に示すようにこの言語では、次のような手順で3次元画像を定義してゆく。

- (1) 必要な属性と基本形状を持った物体を定義する。
- (2) 定義した物体を適当な場所に置く。
- (3) それらを集合演算して目的とする形状を作り、それに属性とエクステントを付加する。必要ならばそれらをさらに組み合わせる。
- (5) まとめて扱いたい物体をひとまとめにじて、適当な位置におく。

このように物体の寸法や属性の係数を可変とすることはできないが、分かりやすい形で階層的に記述できる。

4. プログラミング言語による中間言語の発生

3章で示した3次元画像記述言語を直接使用することによっても比較的容易に3次元画像を記述できる。しかし、物体寸法や属性の係数に具体的な数値を与えてはならないので、例えば部品の寸法や位置を物体全体の大きさにあわせて自動的に決めるといったことができない。また、多数の同じような物体を定義するときは、それらを全て一つづつ記述しなければならない。

表3 C言語による情景定義生成用関数

```

attribute red_plastics
  color 1.0 0.5 0.5
  ks 0.8   sf 50
end

attribute blue_plastics
  color 0.5 0.5 1.0
  ks 0.8   sf 50
end

primitive sph1 sphere 1 end
primitive sph2 sphere 0.9 end
primitive pln zxplane end

construct red_cup (sph1-sph2)*pln
  attribute red_plastics
end

construct blue_cup (sph1-sph2)*pln
  attribute blue_plastics
end

put blue_cup2 blue_cup
  tran 3 2 0
end

group cups red_cup,blue_cup2 end
display cups

```

図2 中間言語による3次元画像記述例

そこで、これを解決するために、この言語による3次元画像の記述を通常のプログラムを実行させることで作成する。これにより、手続き型プログラミング言語の計算機能、引き数受渡し機能、実行制御機能が3次元画像の記述に利用できる。

3章で示した言語ではすでにシンボルを多用して階層的に記述できるようになっているので、この言語による3次元画像記述を発生させるプログラムは非常に簡単なものでよい。例えば、C言語で形状組合せ文を作り出すには図3のような関数を用意すればよい。このように文字列の出力のみで済む。現在我々が用意している3次元画像記述のための関数を表3に示す。表3はC言語の関数であるが、PASCALやFORTRANを使用しても同様なサブルーチンを用意することは簡単なので、この程度の中間言語を定めておけば各種のプログラミング言語から利用できる。

図4に表3の関数群を使って記述したn角柱を生成する関数を、図5に5角柱を生成したときの中間言語への変換結果を、図6に生成された5角柱の画像を示す。この例で分かるようにプログラミング言語のパラ

◆物体定義用関数

```

(1) 属性定義
attribute_begin( name );
color( r,g,b );
ambient( n );
....
attribute_end();

(2) 物体生成
plane( name, a,b,c,d, attr_name );
xyplane( name, attr_name );
....
cone( name, a,b, attr_name );

(3) 物体組合せ
construct( name, expression,
          attr_name, x,y,z,r );

(4) 物体等のグループ化
group( name, name_list,
       attr_name, x,y,z,r );

(5) 物体・グループの移動回転
put_begin( name1, name2 );
tran( x,y,z );
....
zrot( n );
put_end();

(6) 表示物体・グループ指定
display( name_list );

```

◆観測条件定義用関数

```

(1) 光源設定
light( type, x,y,z, r,g,b );

(2) 視点位置指定
from( x,y,z );

(3) 注視点位置指定
to( x,y,z );

(4) 上方向指定
up( x,y,z );

(5) ズーミング指定
zoom( n );

(6) 移動回転指定
tran( name, x,y,z );
xrot( name, n );
yrot( name, n );
zrot( name, n );

```

```

construct( name, expr, attr, x,y,z,r )
char name[],expr[],attr[];
float x,y,z,r;
{ fprintf(outfp,"construct %s %s\n",name,expr);
  if( attr[0] != NULL )
    fprintf(outfp,"attribute %s\n",attr);
  if( r > 0.0 )
    fprintf(outfp,"extent %f %f %f %f\n",x,y,z,r);
  fprintf(outfp,"end\n");
}

```

図3 3次元画像記述関数のコーディング例

メータ受渡し機能や繰り返し機能を活用して汎用的な物体を記述することができる。

また、繰り返し機能を利用して視点位置や物体位置を記述した観測条件を各フレームごとに生成すれば動画が生成できる。

5. MAGICへのインプリメント

以上述べてきた3次元画像記述方法を高速画像生成装置MAGICにインプリメントした。図7に示すように、MAGICはCP/M68KをOSとする制御プロセッサ(CP)に複数のプロセッサエレメント(

```

prism( name, n, w, h, attr )
char name[],attr[];
int n; float w,h;
{
    char expr[256], temp[256], spline[40];
    char bottom[40], top[40], side[40];
    float angle, r, t, tan(), sqrt();
    int i;

    sprintf( bottom, "%s_L", name );
    sprintf( top, "%s_U", name );
    plane( bottom, 0.0,-1.0, 0.0,-h/2.0, "" );
    plane( top, 0.0, 1.0, 0.0,-h/2.0, "" );
    sprintf( expr, "%s%s", bottom, top );

    sprintf( spline, "%s_S", name );
    plane( spline, 0.0, 0.0,-1.0, -w, "" );
    for( i=0; i<n; i++ )
    {
        angle = 360.0*(float)i / (float)n;
        sprintf( side, "%s%d", name, i );
        put_begin( side, spline );
        yrotate( angle );
        put_end();
        sprintf( temp, "%s%s", expr, side );
        strop( expr, temp );
    }
    t = tan( 3.141592/(float)n );
    r = sqrt( w*w*(1.0+t*t) + h*h/4.0 );
    construct(name,expr,attr, 0.0, 0.0, 0.0, r);
}

```

図4 C言語による3次元画像記述例1

```

primitive penta_L plane 0.00 -1.00 0.00 -1.00 end
primitive penta_U plane 0.00 1.00 0.00 -1.00 end
primitive penta_S plane 0.00 0.00 -1.00 -1.00 end

put penta_0 penta_S
yrot 0.00
end

put penta_4 penta_S
yrot 288.00
end

construct penta penta_L*penta_U*penta_0*penta_1*
penta_2*penta_3*penta_4
attribute gold
extent 0.00 0.00 0.00 1.59
end

```

図5 中間言語への変換結果 (LIGHTで記述)

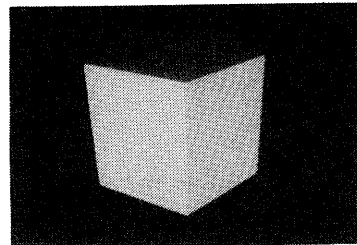


図6 生成された5角柱

PE) が接続されたマルチプロセッサシステムである(3)(4)。現在MAGICにはレイトレーシング法⁽⁵⁾とスキャンライン法の2種類の画像生成ソフトウェアがインプリメントされている。ここで述べた3次元画像記述方法は、レイトレーシング法による画像生成に必要なデータを作成するために使用する。

図8にレイトレーシング法における3次元画像データの構造を示す。このように物体・グループデータ、プリミティブデータ、属性データ、変換行列の4つの表で構成される。画像生成時に深い木構造をたどることをさけるために、物体形状の定義時の階層は圧縮している。つまり、階層的に定義された物体形状は、その物体を構成している全てのプリミティブと集合演算式に変換する。ただし、物体またはグループ間の階層構造は保存し、階層的エクステンツによる画像生成の高速化を図る。

中間言語による階層的な3次元画像の定義は、それを解釈する処理系により上記の各表に直接格納できる具体的な数値データに変換される。中間言語による記

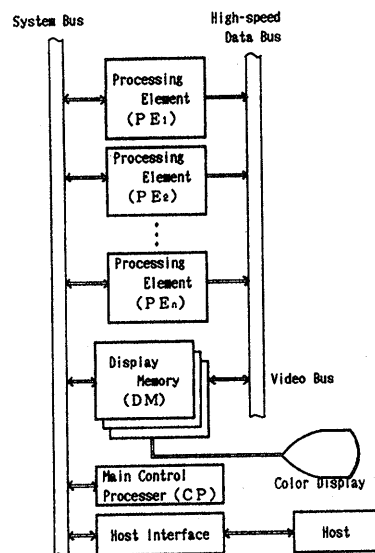


図7 MAGICの全体構成

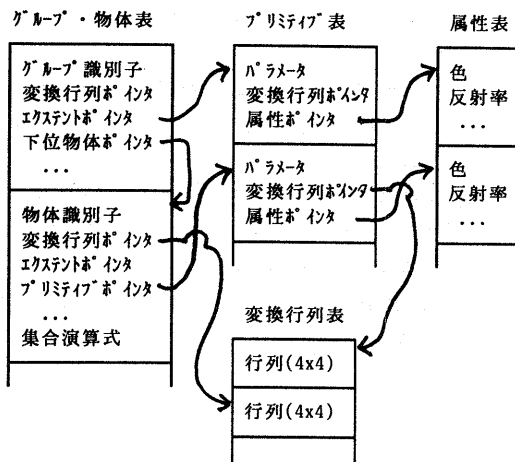


図8 MAGICにおける画像生成用データの構造

述をこのようなデータに変換する処理系はホスト計算機であるVAXとCP/M(制御プロセッサ)上のC言語で作成され、約1100行の大きさである。

MAGICにおける3次元画像データ作成から画像生成までは、以下の手順で行われる。

(1) エディタで3次元画像を定義するCプログラムを作成する。

(2) それをコンパイルし、部品・属性ライブラリとともにリンクする。

(3) プログラムを実行し、結果として中間言語で記述された3次元画像定義を得る。それは一旦ファイルに格納される。

(4) 解釈プログラムにより具体的な画像生成用データに変換し、ファイルに格納する。

(5) 画像生成プログラムはそのデータを読み込み各PEに同時転送して画像生成処理を開始する。観測条件が複数あれば、それを読み込みながら次々と画像を生成する。もし物体位置(変換行列)がその中で指定されていればそれを物体データの変換行列表に入れることにより物体を移動回転させる。

図9にC言語による3次元画像の記述例を、図10に生成された画像を示す。

```
demo( w, r, attr)
float w, r;
char attr[];
{ char cyl[40],bxcyl[40],expr[80],mbxcyl[40];
  int i;

  box( "box", w, w, w, "" );
  cylinder( "cyl", 0.4*w, 0.4*w, "" );

  for(i=0; i<6; i++)
  { sprintf( cyl, "cy%d", i );
    sprintf( bxcyl, "bx%d", i );
    sprintf( expr, "box-%s", cyl );
    sprintf( mbxcyl, "bc%d", i );

    put_begin( cyl, "cyl" );
      translate( 0.0, 0.0, w*(float)i/6 - w/2 );
    put_end();
    construct( bxcyl, expr, attr,
              0.0, 0.0, 0.0, sqrt(1.5*w*w));
    put_begin( mbxcyl, bxcyl );
      translate( 0.0, 0.0, -r );
      yrotate( 60.0*(float)i );
    put_end();
    display( mbxcyl );
  }
}
```

図9 C言語による3次元画像記述例2(主要部分)

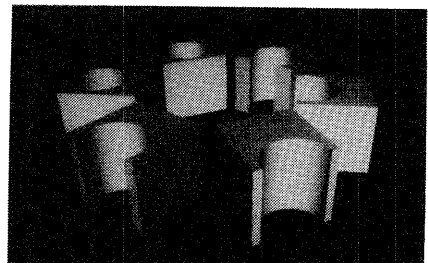


図10 生成された画像

6. あとがき

本文では、3次元画像記述方法の一つとして、3次元画像記述用共通中間言語LIGHTを定め、通常のプログラムによりこの言語で記述された3次元画像の定義を作成し、それを解釈して画像生成用データを作成する方法を提案した。そして、高速画像生成装置MAGICへの実際のインプリメントについて述べた。

本方法は、

(1) 中間言語を生成する副プログラム群を用意すればC、FORTRAN、PASCALなどの各種のプログラミング言語で3次元画像データが作成できる、

(2) プログラミング言語の計算機能、実行制御機能などを利用できるので中間言語は単純なものでよい、などの利点を持つ。

しかし、本文で示した中間言語LIGHTはそれを使用して直接容易に記述できるように定めたものであり、必ずしも最良の記述規則ではない。例えば、形状演算式のような不定個数の物体名を記述しなければならない部分のように、プログラムにより発生させる場合に工夫が必要な部分がある。今後、プログラムにより容易に生成でき、かつ直接記述も容易にできる3次元画像記述中間言語を検討する必要がある。

文 献

- (1) 村上他：“3次元アニメーション用モデリング言語”、日経CG,10月号,PP.146-158,(1986)。
- (2) 福本他：“イメージスコアとその表現方法について”、PIXEL,No.50,PP.109-120,(1986)。
- (3) 三ツ矢他：“高速画像生成装置におけるプロセッサユニットの構成”、グラフィクスとCAD,24-2,(1986)。
- (4) 秋本他：“高速画像生成装置における画像生成ソフトウェア”、グラフィクスとCAD,24-3,(1986)。
- (5) T.Whitted:“An Improved Illumination Model for Shaded Display”, Comm.ACM,Vol.23,No.6, PP.343-349,(1980)