グラフィタスとＣＡＤ　25−4
（１９８７・２・２６）

ＬＩＳＰによるソリッドモデラーの開発

若林哲史，鶴岡信治，木村文隆，三宅康二
三重大学工学部電子工学科

Lispによって書かれた対話形プロトタイプ・ソリッドモデラー PRISM(Prototypal Intaractive Solid Modeler)について紹介する．PRISMは研究室レベルでの研究開発と応用を目的とした，境界表現に基づくコンパクトなソリッドモデラーである．本稿では，PRISMの機能とその実現方法，処理例を示す．PRISMは従来のソリッドモデラーに比べて以下の特色を持っている．
(1)ソリッドモデラーは多量の数値演算とともに，階層的な境界表現とその操作のための複雑なポインター処理を必要とするが，Lispで書くことにより，プログラムが簡潔で明瞭になった．
(2)対話処理のためのコマンドの解釈や実行，ユーザコマンドの定義などが容易にできる．
(3)試行錯誤的なモデリング過程で，中間結果を保存したり，それを消去して記憶領域を再利用したりすることが自由にできる．
(4)Lispで書かれているPRISMは，エキスパートシステムとのリンクやLispマシン上での実行など知識工学，人工知能分野での応用にも適している．

PRISM: Prototypal Interactive Solid Modeler

Tetsushi WAKABAYASHI　Shinji TSURUOKA　Fumitaka KIMURA　Yasuji MIYAKE
Faculty of Engineering, Mie University, Tsu-shi, 514 Japan

We introduce a prototypal interactive solid modeler PRISM written by Lisp. PRISM is a boundary representation based compact solid modeler for laboratory research and applications. In this paper, we describe the function, the implementation of the PRISM and show examples of the modeling process. PRISM has the following characteristics.
(1)Transparency of the Lisp source code without explicit pointer operation for hierachical boundary representation.
(2)Easy definition of the user commands, the interpretation and the execution.
(3)Flexible memory utilization for storing and deleting the intermediate results of the trial and error modeling process.
(4)Easy linkage to the expert system and portability to the lisp machine for the applications in the AI and knowlege engineering field.

1．はじめに

CAD をはじめ，コンピュータグラフィクス，ロボティクス，コンピュータビジョンでコンピュター内に物体や環境のモデルを作り操作するためのツールとしてソリッドモデラー[1][2]がある．最近では，いくつかのソリッドモデラーが市販されているが，市販システムの多くは高価で，気軽に使ってみるというわけにはいかない．また，目的とするシステムのプロトタイプ作りや実験，アプリケーションプログラムやソリッドモデラー自体の開発・研究といった点から考えると，ソース・プログラムが入手でき，使用目的に応じて改造できることが，むしろ必要である．

本稿で紹介する PRISM (Prototypal Interactive Solid Modeler) は，研究室レベルでの開発と応用を目的とした，境界表現に基づくソリッドモデラーでLispにより書かれている．

ソリッドモデラーは集合（ブール）演算などの実行時に多量の数値計算を必要とするので処理効率の観点から Fortran や Pascal で書かれたものが多いが[3]，我々は，大学の研究室における継続的な研究開発・蓄積のために開発効率やアルゴリズム，データ構造の分かりやすさを重視して，Lisp による開発を試みた．ソリッドモデラーは多量の数値計算とともに，階層的な境界表現とその操作のための複雑なポインター処理を必要とするが，Lisp で書くことにより，そのかなりの部分を言語処理系に任せることができプログラムが簡潔明瞭になる．また，ソリッドモデラーの重要な構成要素である対話処理のためのコマンドの解釈や実行，ユーザーコマンドの定義なども Lisp インタープリタの機能が利用できるので，マンマシンインターフェースの設計・開発の労力が軽減される．試行錯誤的なモデリング過程で一時的に出来た不要な物体の表現のために使われている記憶領域の再利用なども Lisp のガーベッジコレクターに任せられ，中間結果を保存したりそれを消去して記憶領域を再利用することも自由にできる．この他，Lisp で書かれたソリッドモデラーは，エキスパートシステムとのリンクや Lisp マシン上での実行など知識工学，人工知能分野での応用にも適していると考えられる．

PRISM は Unix 上の Franz Lisp で実現されたコンパクトなシステムで移植性も高い．

以下では，PRISM の機能，実現と処理例等について紹介する．

2．　PRISMの機能

2－1．　PRISMの概略

PRISM はミニコン上で動作し，表示は，１６７７万色表示可能なグラフィック・ディスプレイを用いて実現されている．プリミティブ発生関数によって発生したプリミティブ・ソリッドや，マウスによって入力した多角形をスイープして得られるソリッドに対して座標変換や UNION, INTERSECTION, DIFFERENCE 等の集合演算関数を施して必要なソリッドを対話的に表示，作成した後，外部ファイルに格納できるようになっている．

本研究では最初に発生される球，直方体，多角柱および多角錐等の基本的な物体をプリミティブ・ソリッドと呼び，それらに演算を施して得られたソリッドとプリミティブ・ソリッドを総称して単にソリッドと呼ぶことにする．PRISM で扱うことのできるソリッドは現在のところ正則な[4]平面多面体に限られている．

システムの構成を以下に示す．

ホスト・コンピュータ	：	microVAX I
ＯＳ	：	ULTRIX-32m
ターミナル	：	PC-9801VX
ディスプレイ	：	NEXUS 6400
		512×480 1677万色
使用言語	：	Franz Lisp, 一部　C

2－2．システムの機能

PRISM の大部分の機能は，入力となるデータを引数として受け取り，演算結果のソリッドを値として返す関数の形で用意されている．

現在用意されている関数は大きく分けて次の５つに分類される．

ａ．プリミティブソリッドの発生
ｂ．スイープ
ｃ．ソリッドに対する幾何学的変換
ｄ．ソリッドに対する集合演算
ｅ．ソリッドの表示
ｆ．外部ファイルとのソリッドの入出力
ｇ．その他

関数名と機能を以下に示す．

ａ．プリミティブソリッド発生関数

box()　　：立方体の発生

tetra()　:正四面体の発生
cylinder() :円柱の発生．３２角柱で近似
cone()　:円錐の発生．３２角錐で近似
sphere()　:球の発生．緯度方向に８分割，経度方向に１６分割して多面体近似
sphere2(n) :球の発生．緯度方向にｎ分割，経度方向にｎ×２分割して多面体近似
prism(n)　:ｎ角柱の発生．
pyramid(n) :ｎ角錐の発生．
torus(n r1 r2) :トーラスの発生．小半径r2と大半径r1で指定される近似円環体．小半径r2で指定される円断面をｎ角形で近似し，経度方向にはｎ×２分割する．
ただしr1≧r2とする．

b．スイープ

sweep(f x y z) :多角形 f をx,y,z各方向にsweepさせたソリッドを発生．
revolve1(f n)　:３次元空間においてx-y平面上にある多角形 f を，y軸のまわりに回転させた回転体を発生．ｎは回転方向の分割数．
revolve2(v-list n) :３次元空間においてx-y平面上の点列を結びy軸の回りに回転させた回転体を発生．ｎは回転方向の分割数．点列の始点と終点は回転軸上にある．

c．ソリッドに対する幾何学的変換

trans(s x y z) :ソリッドsを各方向にx,y,zだけ移動する．
rotx(s thetax) :ソリッドsをx軸まわりにthetax回転する．
roty(s thetay) :ソリッドsをy軸まわりにthetay回転する．
rotz(s thetaz) :ソリッドsをz軸まわりにthetaz回転する．
scale(s x y z) :ソリッドsを各方向にx,y,z拡大する．

d．ソリッドに関する集合演算

comp-s(s)　　:ソリッドsの補元
int-ss(s1 s2)　:２つのソリッドs1,s2の集合積
uni-ss(s1 s2)　:２つのソリッドs1,s2の集合和
　　comp-s(int-ss(comp-s(s1) comp-s(s2))
diff-ss(s1 s2)　:２つのソリッドs1,s2の集合差
　　int-ss(s1 comp-s(s2))
append-ss(s1 s2) :２つのソリッドs1,s2の連結．s1,s2に共通部分のない場合集合和となる．

e．ソリッドの表示

disp(s)　:ソリッド s のすべての多角形をワイヤーフレーム表示する．
disp2(s) :ソリッド s が凸多面体の場合のみ隠面消去をしてワイヤーフレーム表示をする．
disp3(s) :ソリッド s を重ね塗りにより隠面消去してシェーディング表示する．
disp-s(s):ソリッド s をZバッファ・アルゴリズムを用いてシェーディング表示する．

f．ソリッド・データの入出力

dsave(s 'sname) :ソリッドsの内部表現をディスクにsave
dload('sname)　:ソリッドの内部表現ををディスクからload

g．その他

copy-s(s)　: ソリッドのコピー

3．データ構造とアルゴリズム

3－1．　ソリッドの内部表現

境界表現は，ソリッドをその境界である多角形(polygonal face)の集合で表し，多角形はその境界である辺(edge)の集合で表し，辺(edge)はその境界である頂点(vertex)の対で表す．PRISMでは，これらを直接的に表現する入れ子になったリストをソリッドの主要な内部表現として用いている（図１）．但しソリッドはそのソリッドが有界か有界でないかを示すための符号と境界リストの対で表されている．符号はtとnilの値をとり，nilはソリッドが有界であることを，tはソリッドが有界でないことをそれぞれ表す．PRISMにおいてソリッドを表現するこのようなリストのことを以後エッジリスト表現と呼ぶことにする．

頂点はLispのシンボル（Fortranなどの変数に相当）

で表され，x，y，z座標の値を属性リスト内に持っている（図２）．頂点を表すシンボルは，ソリッドの発生などによって新たな頂点を発生する必要が生じた時にシステムによって自動的に発生される．ソリッドの幾何学変換は，ソリッドを構成するすべての頂点に対して，そのシンボルの属性リストにある座標値を変更することによって実行される．

多角形を表す辺の集合において各辺は，物体の外部から内部に向かって多角形を見たときその内部を右手にみる向き，つまり物体の内部に向かって右回りの向きを持っている．辺の順序は問わない．辺の向きは次の例のように頂点の順序で示される．

V1 から V2 に向かう辺 ： (V1 V2)

V2 から V1 に向かう辺 ： (V2 V1)

この方法では，図１の多角形 P1 は，

((V1 V2)(V2 V3)(V3 V4)(V4 V1))

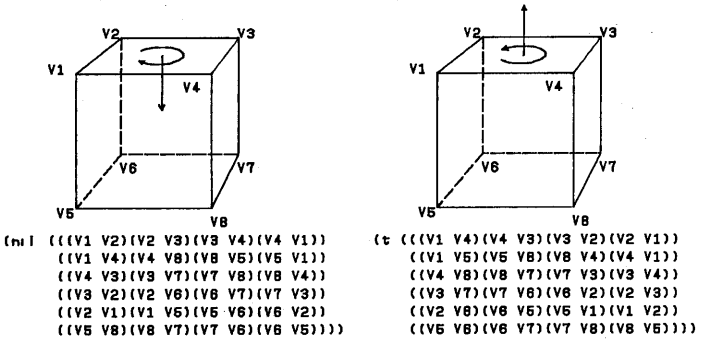
のように表現される．他の方法として P1 を

(V1 V2 V3 V4)

と表現した方が簡潔であり，処理量や記憶領域を節約できるが，処理の容易さ一様性を重視してここでは前者の方法を取った．この方法によると，複数の多角形や穴を持っている多重連結多角形も同様に表現でき，辺の順序を問わないため処理が簡単になる．

図３にエッジリスト表現のメモリー上での様子を示す．

エッジリスト表現は，ソリッドモデラーでよく用いられるウィングドエッジ表現などに較べると非常に単純であるが，ソリッドをあい昧さなく表現することができ，一方から他方へ変換できるという意味で両者は等価である．また処理アルゴリズムもエッジリスト表現のほうが単純になる．Putnam[5] は n 次元物体の集合演算を実行する一般的なアルゴリズムを述べているが，内部表現として一種のエッジリスト表現を用いている．また集合演算以外の処理では，処理効率の面でもほとんど問題ない．　　しかし，図３に示されたポインターの様子からもわかるように，エッジリスト表現では辺や面の隣接関係がウィングドエッジ表現でのように直接的に表現されていないので対象の一貫性（Object Coherencey）を利用して集合演算における交点（線）計算などを必要最小限にとどめるのが難しい．しかしこれらの表現方法の比較や，併用に関する検討は今後の課題とし，今回はLispでの取扱が容易なリストにより直接実現できるエッジリスト表現を用いることにした．



(a)有界なソリッド　　(b)有界でないソリッド

図１　データ構造の例

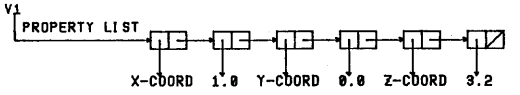


図２　座標データを頂点の属性リストで表現した例のbox-and-arrow図

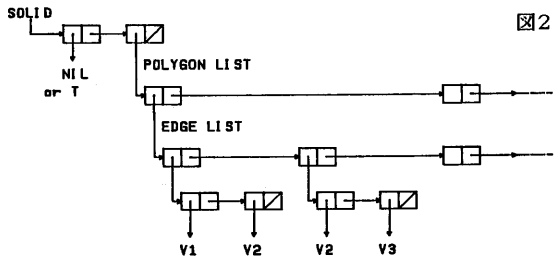
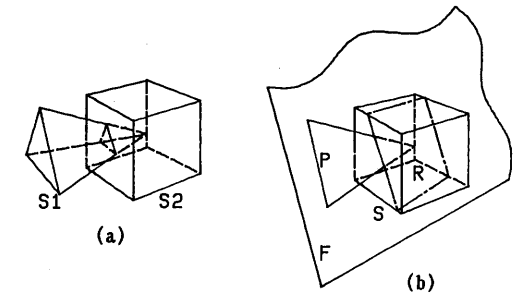


図３　ソリッドのデータ構造のbox-and-arrow図



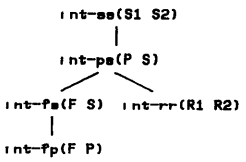


図４　int-ssの構成

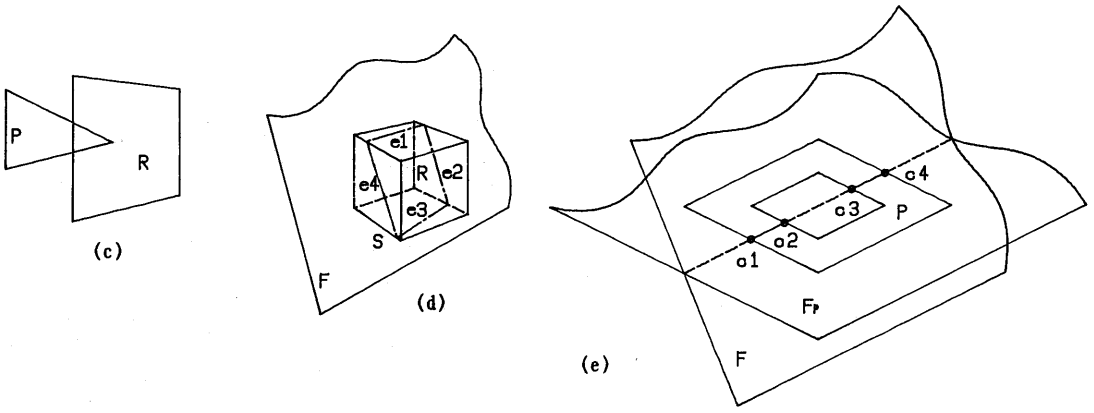


図５　int-ssのアルゴリズム

３－２．処理アルゴリズム

PRISMにおける集合演算のアルゴリズムの概略について述べる．二つのソリッドA，Bに対する集合演算として，集合和A＋B，集合積A×B，集合差A－Bがよく用いられる．これらは集合積のアルゴリズムが有界でないソリッドに対しても適用可能なら，補元と集合積とで実現可能である．そのためまず最初に補元をとる関数 comp-s と集合積をとる関数 int-ss を作成し，他の２つは，ドモルガンの法則

集合和　uni-ss(A B)
　　= comp-s(int-ss(comp-s(A) comp-s(B)))
集合差　diff-ss(A B)
　　= int-ss(A comp-s(B))

を用いて実現した．

comp-s は単に，与えられたエッジリスト表現内のエッジの向きを逆にし，符号を反転したエッジリストを関数の値として返す．

int-ssは図４のように階層的な関数呼び出しの形で実現されている．以下では図４に従いint-ssのアルゴリズムの概略を述べる．集合演算のアルゴリズムについては[5]の記述がわかりやすく一般的である．以下のアルゴリズムも同様の考えに基づいたものであるが，３次元，２次元の集合積を個別に実現している点が違っている．また，２つのソリッドの面が接している場合の処理については説明を省略する．

int-ss(S1 S2)：２つのソリッドの集合積を求める．

図５(a) を例にとると，三角錐S1と立方体S2の集合積は三角錐の先端のS2に食い込んでいる部分である．まずS1に注目し，S1の境界を構成しているすべての多角形に対してS2との集合積を求める．この多角形とソリッドの集合積を求める関数が，int-ps である．int-ps は結果が空の場合nilを返す．同様のことをS2のすべての多角形に対しても行い，得られた結果をリストにすればS1×S2の境界リストが得られ，それに符号を付けたものを int-ss の結果とする．　符号はS1とS2の符号の論理和である．

int-ps(P S)：多角形PとソリッドSの集合積，つまりPのうちSの内部に含まれる部分を求める．　図５(b)

を例に取ると，まずある多角形Ｐがのっている平面(surface)ＦでＳを切った切口ＲをInt-fsで求め，図５(c)に示す様に２次元の集合積アルゴリズムに帰着してＰとＲの積を求める．　２次元の集合積の計算は，int-rrが行う．

int-fs(F S)：平面Ｆで切ったソリッドＳの切口を求める．図５(d)では平面Ｆでソリッドを構成する各多角形をint-fpを用いて切り，結果e1～e4をリストにすれば切口の多角形Ｒが求められる．

int-fp(F P)：平面Ｆで切った多角形Ｐの切口をエッジリストとして返す．Ｐが複雑な凹多角形や複数の多角形から成り立っている場合，このエッジリストには複数のエッジが含まれることになる．図５(d)の例ではint-fp(F P)の結果はすべて一つのエッジになるが，さらに一般的な場合を図５(e)に示す．図８で多角形Ｐを構成している各エッジと平面Ｆとの交点c1～c4を求めれば，これらの交点はすべて平面Ｆと，Ｐを含む平面F_pの交線上に存在する．この交点列を外積（Ｆの法線×F_pの法線）方向に昇順ソートし，２つずつ取ってエッジを作りリストにすれば結果として((c1 c2)(c3 c4))が得られる．この外積方向のソートにより，平面Ｆでソリッドを切った切口は，Ｓが有界なソリッドの場合はＦの法線方向に向かって右回りに，Ｓが有界でないソリッドの場合左回りに求められるため，有界でないソリッドにたいする演算も正しく行われる．

int-rr(R1 R2)：２次元で領域(region)Ｒ1とＲ2の集合積を求める．領域は３次元の多角形を２次元に投影したものである．

これらの操作の各段階で，明らかに共通部分のないものに対する無駄な処理を省くのに box-check を行っている．

４．処理例

PRISM における対話処理の例として，六角柱を発生させそのコピーを回転，平行移動したものと，もとの六角柱との UNION をとる様子を以下に示す．

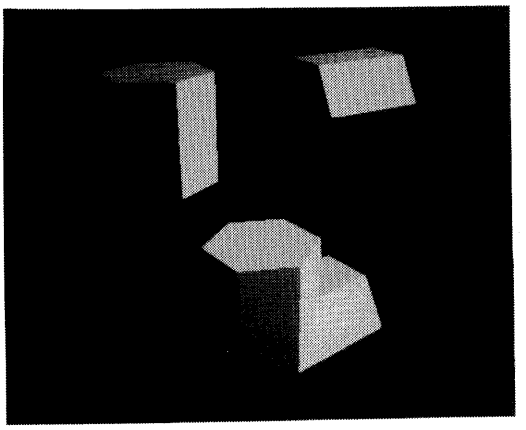
```
>(setq s1 (prism 6))
```
六角柱を発生させ s1 に代入する．

```
>(setq s2 (scopy s1))
```
s1 をコピーして s2 に代入する．

```
>(trans (rotx s2 90.0) 1.0 -1.0 0.0)
```
s2 を回転，平行移動する．

```
>(setq s3 (uni-ss s1 s2))
```
s1 と s2 の union をとって s3 に代入する．

```
>(disp3 s3)
```
s3 を表示する．



以上の例では s1，s2 が中間結果として保存されている．次のように入力しても同じ結果が得られるが，この場合は中間結果が保存されない．但しこの例では，最初の六角柱をコピーする代わりにソリッド発生関数 prism を２度使っている．

```
>(setq s3 (uni-ss (prism 6)
                 (trans (rotx (prism 6) 90.0)
                        1.0 -1.0 0.0)))
```

この例からもわかるように，PRISM では，ソリッドを作成する Lisp の Ｓ－式 (Symbolic Expression) そのものが CSG 表現となっている．

他の例を図６～８に示す．

図６はローテーショナル・スイープによって発生させた回転体である．図７はローテーショナル・スイープを行いながら平行移動することによって発生させたバネである．図８はリニア・スイープと集合演算により作成した歯車の組合せである．

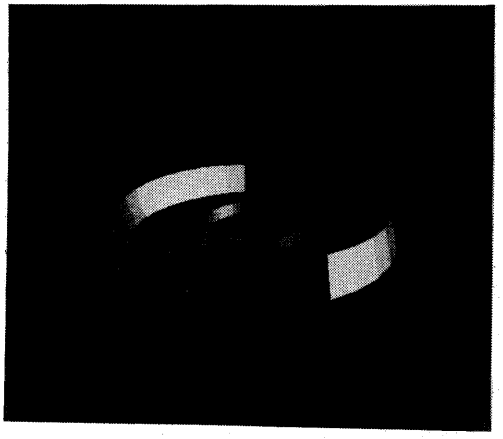


図6　ローテーショナル・スイープにより発生した回転体

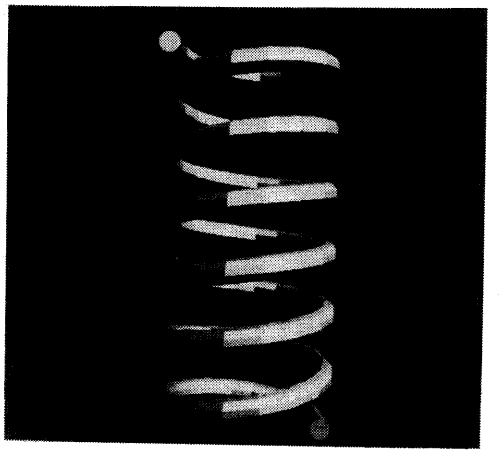
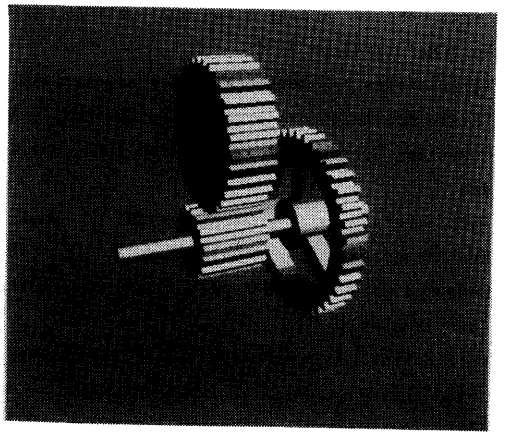
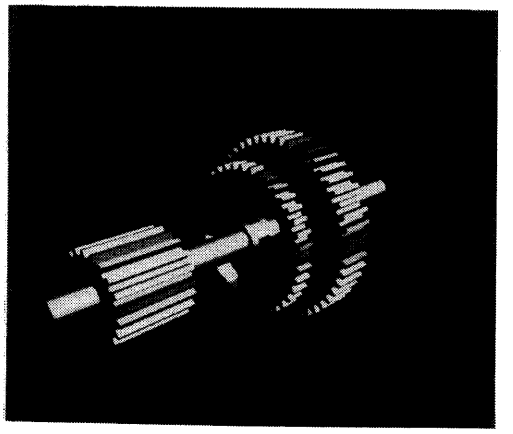


図7　ローテーショナル・スイープと平行移動によるバネ



(a)



(b)

図8　リニア・スイープと集合演算による歯車の組合せ

5．むすび

Lispによって書かれたプロトタイプ・ソリッドモデラー　PRISMについて紹介した．　開発の過程を通してLispによる開発の得失として以下のことがわかった．

(1)Lispの対話的プログラミング環境は，人工知能システムなどの複雑で大きなシステムの研究・開発に適しているといわれている[6]．PRISMの開発においても対話的プログラミング環境やステッパ，トレーサなどの完備したデバッグ・ツールの活用によって開発効率をあげることができた．

(2)Lispのリスト処理，記号処理の機能を利用することによって，直接的なポインター操作のプログラミングから開放され，プログラムも簡潔になった．一般にポインターは，プログラマーにとって扱い易いものではなくプログラムを難解なものにし，バグなどの原因になりやすいため C などを除く多くの高級言語では直接扱うことが制限されている．また配列の添字でポインターをシミュレートする場合には処理効率も低下する．

(3)対話的モデリングのためのコマンドの解釈や実行，ユーザーコマンドの定義などに Lisp インタープリタの機能を利用することで，マン－マシンインターフェースの設計・開発の労力が軽減された．

(4)PRISMのソースプログラムサイズは約２２００行である．ポインター操作やマン－マシンインターフェイスのためのソース・コードが不用のため他言語による実現に較べてかなりコンパクトになったと思われる．

(5)Lispの動的メモリー管理機能は，試行錯誤的なモデリング過程で一時的に発生したりコピーされた物体に使われている記憶領域の再利用などに適している．

(6)ソリッドを作成するためのLispのＳ－式そのものがコンパクトなＣＳＧ表現となっている。

(7)Lispで書かれたソリッドモデラーはエキスパートシステムとのリンクや Lisp マシン上での実行など知識工学，人工知能分野での活用にも適していると考えられる．

一方，Ｌｉｓｐによる実現の問題点としては，処理効率の低下があげられる．現在４．の処理例の集合演算に約５分かかる．これは，他のモデラーに較べ１０倍以上の時間であるが[7]，そのうち何割がLispによる実現によるものか現時点では不明である．プログラムの実行時間には，データ構造，アルゴリズム，プログラミング言語，ハードウェアの性能等が関係するが今回はいずれも高速化を配慮して選択されていないからである．データ構造，アルゴリズムについては現在，ウィングドエッジ表現の一種であるハーフエッジ表現をオイラーオペレータで操作する高速版を作成中である．また，使用した　micro VAX I　には浮動少数点演算装置がなく数値計算に８０８７を使用したパソコンに較べても十数倍の時間がかかるので，他の計算機に移植を検討中である．研究開発用のソリッドモデラーとしては，Fortran,Cなどで実現したものに較べてある程度の処理効率の低下ならLispの他のメリット考えれば許容できると考えている．

謝辞

本研究に関し，日頃御指導賜る名古屋大学　横井助教授，並びに熱心に御討論下さる研究室の皆様に深く感謝いたします．

文献

(1)A.A.G.Requicha, H.B.Voelker: "Solid Modeling :Current Status and Research Directions," IEEE Computer Graphics and Applications, Vol.3.No.7, Oct. 1983,pp.25-37

(2)図形処理情報センター，”３次元形状モデラーの構造”，PIXEL No.24 Dec.1984

(3)A.A.G.Requicha, H.B.Voelker: "Solid Modeling: A Historical Summary and Contemporary Assesment, "IEEE Computer Graphics and Applications, Vol.2. No.2,Mar. 1982,pp.9-24

(4)Aristides A.G.Requicha: "Representations for Rigid Solids:Theory,Methods,and Systems," ACM Computing Surveys,Vol.12.No.4,Dec.1980, pp.437-464

(5)L.K.Putnum, P.A.Subrahmanyam: "Boolean Operations on n-Dimensional Objects, "IEEE Computer Graphics and Applications, Vol.6.No.6, Jun. 1986,pp.43-51

(6)P.H.Winston, B.K.P.Horn:"LISP,"Addison-Wesley Publishing Company, Inc., Reading,Mass., U.S.A.

(7)Fujio Yamaguchi, Takaomi Tatemichi, Ryoji Ebisawa: "Applications of the 4*4 Determinant Method and the TRIANGLE PROCESSOR to Various Interference Probrems, "Advanced Computer Graphics ,Springer-Verlag Tokyo 1986