グラフィクスとCAD 52－5
（1991. 8. 16）

関数形式による樹木モデリングシステム

春口 巌

日本大学大学院理工学研究科

種々の形状のモデリングの際に、樹木など複雑な3次元形状を容易に、かつ迅速にモデリングするための手段として、モデリングという手続きをプログラム化する方法がある。モデリング手続きを制御するには、あらかじめ定めた、形状を定義するのに有効なパラメータを編集する方法をとり、編集結果の保存方法としては、基本的にパラメータのみを保存する。こうすることにより、モデリング手続き自身の簡略化とモデリングデータを保存する場合の計算機資源としての記憶装置の節約を図ることが出来た。

本報告は、モデリング手続きをプログラム化した樹木生成関数に、ユーザー・インターフェイスを備え、関数形式のモデリングシステムとして構築したシステム事例を紹介するものである。

Functional Tree Modeling System

Iwao Haruguchi

College of Science and Technology, Nihon University

1-8 Kandasurugadai Chiyoda-ku, Tokyo 101, Japan

Among various modeling method, there is a functional or procedural modeling method. This report writes about the functional tree modeling system which enables easy and speedy modeling. This system provides user with the interface to edit just parameters to control modeling procedure and output shape of models. Basically, users have to save parameters after modeling. That is to say, the set of parameters is the expression of shape information. Apparently, the size of parameter file is less than that of polygon file. This fact implies that user can save disk resources.

１．はじめに

３次元ＣＧシステムにおいて、画像を生成する際には、３次元モデルを作成し、その後でレンダリングを行うが、多くの場合、モデリングの作業には次の２つの問題がつきまとう。

１．精巧なモデルを作成するには、多大な時間がかかる。
２．表現するべき物体の数が多いとデータが膨大になり、コンピュータ資源である記憶装置を消費する。

樹木の例ではないが、図１、図２はパワーボートとヨットクルーザーをモデリングしたものである。これらをモデリングするために、プロフェッショナルのＣＧ制作者が費やした時間は、各々約２０時間と１０時間である。画像の制作過程において、モデリング作業自体が、本質的な作業でない場合、－　たとえば景観シミュレーション・システムにおいては、画像の構成要素としての構造物のモデリング作業よりも構造物の配置作業に試行錯誤を要し、また、配置作業がより本質的な作業である。－　モデリングに多大の時間を消費することは本質的な問題となる。

また、詳細なモデリングを行った場合、３次元形状を具体的に表現したデータ　－　たとえば、ポリゴンファイル　－　は非常に大きなサイズになり、画像に表現するべき物体の数が多い場合には、データファイルの大きさは、コンピュータの資源であるディスクスペースを節約するという観点からみて、好ましいことではない。

以上の２つの問題にたいして、１つの解決策、あるいは１つの解決の方向性を示すものとして、筆者の開発した関数形式の樹木モデリングシステムを紹介したいと思う。

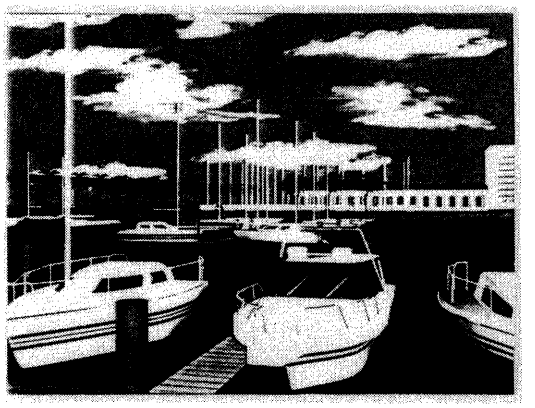


図１

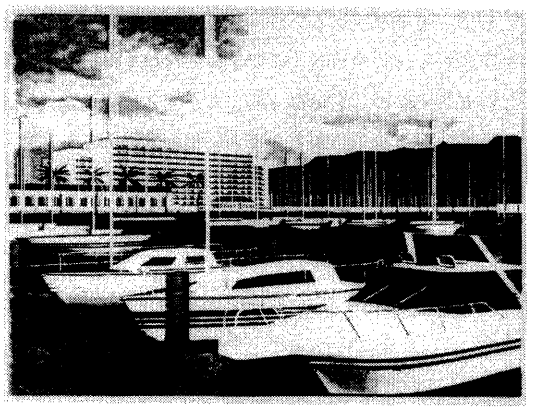


図２

２．関数モデリング

さまざまなモデリングの方法が試みられてきているが、関数モデリングという概念を整理した形で、また一般的な形で論じているのは、［１］であろう。

すなわち、関数という概念の本質が、入力に対して何らかのプロセスを施し、対応する出力を得ることであるから、入力として　－　樹木の場合は、枝の分岐角度、枝の長さの収縮率、枝の太さ、枝の太さの収縮率、葉の数や葉の大きさなど　－　をとり、出力として３次元形状をポリゴンで表現したものをとるならばこれも関数と呼べるであろう。また、この関数によって従来のモデリングという手続きがなされたのであるから、この関数によるモデリングは関数モデリングと呼ぶことが出来るであろう。

Input

枝の分岐角度など
のパラメータ

⬇

Function

関数形式による
モデリング

⬇

Output

樹木形状を表現した
3次元モデル

3．システム具体例

当システムでは、次に挙げる3種類の型に生成アルゴリズムを分けて生成関数を作成した。

各関数に設定されるべきパラメータのうち、形状制御情報パラメータ（たとえば、枝の分岐角度など）は各構造物生成関数によって固有のパラメータを持っているが、当システムのすべての構造物生成関数に共通するパラメータが存在して、それは表現精度と呼ばれるものである。具体的には、当システムでは、ポリゴンの集合体として、形状を表現するので、形状の近似精度を表現精度とする。これは、すなわち樹木の幹について言うならば、断面近似形状をn角形とすると精度が高い場合にはnの値をを大きくし、粗い精度の場合にはnの値を小さくすることが出来るようになっている。

－ 内蔵関数 －

1． L－System型
2． 低木型
3． 特殊樹型

3．1 樹木の種類

ここではまず、関数モデリングをどのように具体化したかをL－System型の樹木を例にとって述べたい。

3種類の樹木型のそれぞれのアルゴリズムが表現する形状は以下のとうりである。

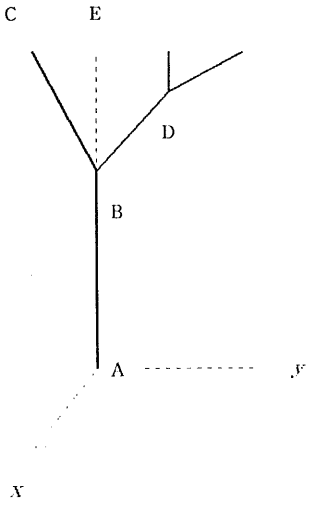
❶ L－System型 ： いわゆる3次元L－Systemに基礎を置くもので、桜の木など。
❷ 低木型 ： 人工的に刈り込んだ低木。
❸ 特殊樹型 ： ヤシなどの特殊な形状の木。

以下に、❶ L－System型の形状生成アルゴリズムの概要を説明しようと思うが、このアルゴリズムは、基本的には［2］に依っているので、詳細はこの論文を参照されたい。
（［2］ではA－Systemと呼んでいる。）

L－System型の樹木生成関数の入力パラメータは以下に述べるものが主なものである。

・根元の幹の開始点と終了点
・枝の分岐角度の最大値と最小値
・枝の分岐角度の乱雑さの度合い。
・枝の縮小率。
・枝の分散角
・枝の太さ
・枝の太さの縮小率
・葉の多さ
・葉の種類

パラメータについて



A，B　：　根元の幹の開始点と終了点

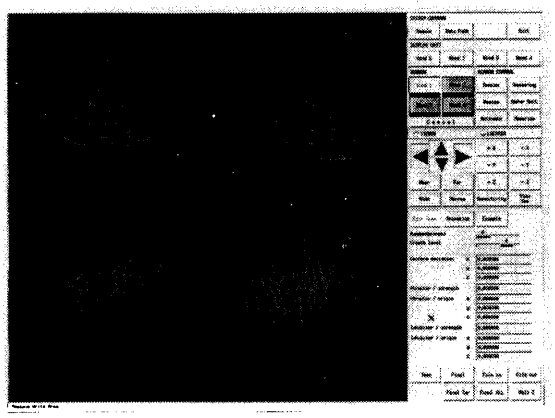
∠EBC，∠EBD　≦　枝の分岐角度の最大値
∠EBC，∠EBD　≧　枝の分岐角度の最小値

R_1　＝　BC／AB　（枝の縮小率）
R_2　＝　BD／AB　（枝の縮小率）

枝の分散角　α　＝　∠ⒸAⒹ
（但し、Ⓒ、Ⓓはそれぞれ C，D をｘｙ平面に投影した点）

次に関数の内部アルゴリズムについて述べる。Ｌ－Ｓｙｓｔｅｍ型の樹木生成関数のアルゴリズムの骨格は、再帰呼出しであり、疑似コードで表現するならば以下のようになる。

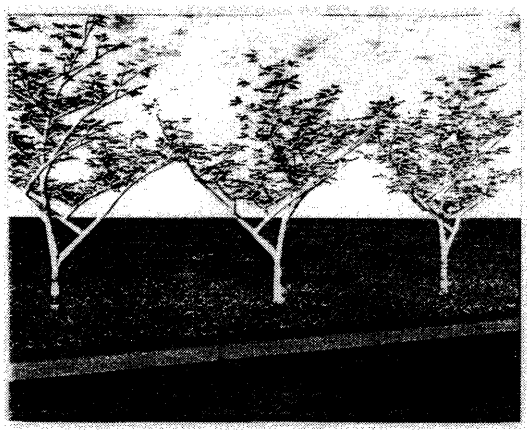
```
FUNC MakeTree(BranchInformation)
INPUT
  BranchInformation   Information of
                      Parent Branch
DATA
  ARRAY ChildBranches(2) of the same
        type as BranchInformation is
BEGIN (* Make branches )
   Calculate 2 child branches
    acording to the information written
    in BranchInformation.
   Give level value to the child branches,
    which value is BranchInformation's
    level - 1
   IF NOT the level of child branches is 1
      NULL
   ELSE(* Make successors of child
          branches *)
      MakeTree(ChildBranches(1))
      MakeTree(ChildBranches(2))
   ENDIF
END
```



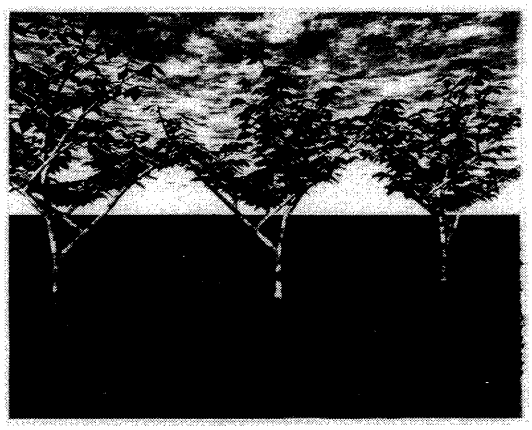
ユーザー・インターフェイス

すなわち、関数 MakeTree は、ある枝の情報を受け取ったら、この枝の先端から２～３本の子枝を生成させる。そこで、生成した枝のレベルは受け取った枝情報に書かれているレベルから１引いたレベルを設定する。この時、生成した枝のレベルが最も先端を示すレベルでなければ、さらに、今計算した子枝に関数MakeTree を作用させる。

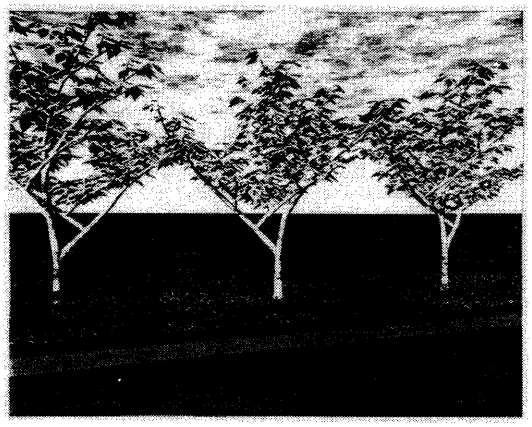
関数 MakeTree の受け取るパラメータである BranchInformation には、枝の始点と終点の座標やレベルなどが書かれている。



春



夏



秋

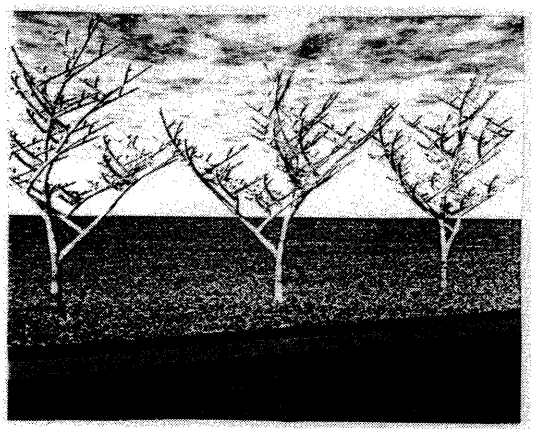
出力は、以下に記すようにポリゴン・ファイルとなっている。但し、このシステムを組み込むシステムの要求仕様により、出力されるポリゴン・ファイルの形式は、構造物ポリゴン・オブジェクトという形式で出力される。（C言語の構造体を連続的に記述する形式）

－　構造物オブジェクト構造体　－

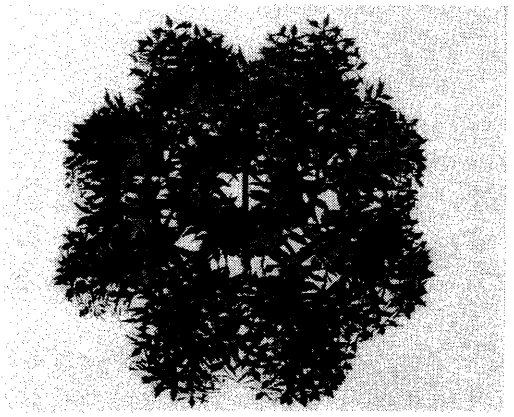
```
typedef struct {
  char  com[SizeLABEL];  /* コメント */
  int   obj;    /* オブジェクトラベル */
        /* FdmSTRUCTURE_P  :  ポリゴン形式の
           構造物オブジェクト */

  char  anim[SizeLABEL];
        /* アニメーションラベル */
  float min_t;
        /* アニメオブジェクト使用開始時刻 */
  float max_t;
        /* アニメオブジェクト使用終了時刻 */
  float cx;
        /* 拡大・縮小の中心位置　X座標値 */
  float cy;
        /* 拡大・縮小の中心位置　Y座標値 */
  float cz;
        /* 拡大・縮小の中心位置　Z座標値 */

  int   nv;      /* 頂点データ数 */
  int   np;      /* ポリゴンデータ数 */
  FdmStructurePolygonObjVertex  *vertex;
  /* 頂点データ構造体の配列へのポインター */
  FdmStructurePolygonObjPolygon *polygon;
  /* 面データ構造体の配列へのポインター */
}       FdmStructurePolygonObj;
```

この構造体は、ポリゴンの標準的な表現方法である頂点リストと面リストの組に対して、それぞれのポインターを含んでいる。頂点データ構造体と面データ構造体の具体的なフォーマットは次ページ以下に掲載するとうりである。

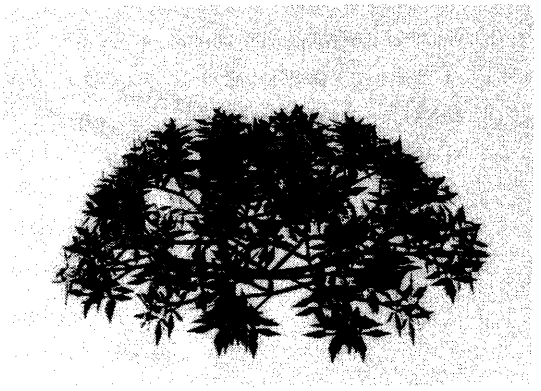
－　頂点データ構造体　－



冬

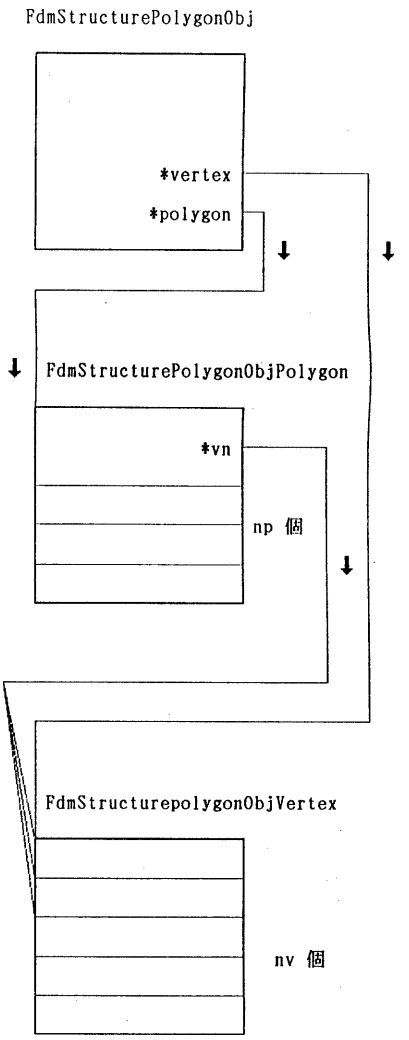


低木　－　上から見た図



低木　－　斜め上から見た図

```
typedef struct {
  int   vn;     /* 頂点番号 */
  float vx;     /* Ｘ座標値 */
  float vy;     /* Ｙ座標値 */
  float vz;     /* Ｚ座標値 */
  float nx;     /* 法線Ｎｘ -1.0≦Nx≦1.0 */
  float ny;     /* 法線Ｎｙ -1.0≦Ny≦1.0 */
  float nz;     /* 法線Ｎｚ -1.0≦Nz≦1.0 */
} FdmStructurePolgonObjVertex;
```



－ 面データ構造体 －

```
typedef struct {
  int   pn;      /* ポリゴン番号 */
  int   type;    /* FdmTR_POLY  :  三角形 */
                 /* FdmQD_POLY  :  四角形 */
  FdmStructurePolygonObjVertex  *vn[4];

  char  texa[SizeLABEL];
                 /* 材質ラベル（表面）*/
  char  texb[SizeLABEL];
                 /* 材質ラベル（裏面）*/

  int   switcha;
        /* 表面マッピングの有無
           FdmMAP_ON / FdmMAP_OFF */
  float ua[4];
        /* 各頂点のマッピングｕ座標
           ０．０  ～  １．０  */
  float va[4];
        /* 各頂点のマッピングｖ座標
           ０．０  ～  １．０  */
  float wa[4];
        /* 各頂点のマッピングｗ座標
           ０．０  ～  １．０  */
  int   switchb;
        /* 裏面マッピングの有無
           FdmMAP_ON / FdmMAP_OFF */
  float ub[4];
        /* 各頂点のマッピングｕ座標
           ０．０  ～  １．０  */
  float vb[4];
        /* 各頂点のマッピングｖ座標
           ０．０  ～  １．０  */
  float wb[4];
        /* 各頂点のマッピングｗ座標
           ０．０  ～  １．０  */
}       FdmstructurePolygonObjPolygon;
```

３．２その他

ここでは、３．１で触れなかった当システムに含まれるＬ－Ｓｙｓｔｅｍ型の樹木以外のモデリング関数について若干述べておく。

［２］で紹介される３次元Ｌ－Ｓｙｓｔｅｍでは、かなりの種類の樹木を生成できるが、人工的に刈り込んだ樹形には向かない。しかし、景観シミュレーションなどの場合は街路樹や高速道路の中央分離帯の低木など人工的に刈り込んだ形の樹木がどうしても必要となる。また、リゾート開発などにおける景観シミュレーションでは、ヤシの木の類が必要となる場合が多くこれもまた、Ｌ－Ｓｙｓｔｅｍ型の樹形とは異なる形状をしている。そこで、低木型と特殊樹型を新たに作成したのであるが、これらの形状生成アルゴリズムがＬ－Ｓｙｓｔｅｍ型のそれと本質的に違う点は、地面から垂直に伸びる幹を他の枝とは別扱いにしたことである。

４．利点と欠点

利点を箇条書きにすると次のようになる。

① モデリング作業の簡略化。
パラメータを編集するだけなのでモデリングに要する作業時間の短縮を図ることができる。

② コンピューター資源の節約。
モデリング・データをパラメータ形式で保存すれば、ディスク・スペースの節約になる。

また、欠点を箇条書きにすると次のようになる。

① モデリング作業が簡略化されたので、膨大なモデリング・データを比較的楽に作成することができてしまう。（レンダリングの直前にポリゴンデータに展開した時）

② 独創的な形状生成ができるわけではない。

③ 現在、テクニカルな値をパラメーターとして持っているので、一般のＣＧ製作者が編集をするためには慣れるまでに時間がかかる。

５．将来への展望など

まず、行わなければならないのは、さらに扱いやすいユーザー・インターフェイスの整備であり現在改良中である。また、扱う構造物の種類を樹木だけに限らずに種々のモデリング関数を搭載し、さらにＣＧ製作者の使えるシステムに改良したいと考えている。

将来的には、アニメーションへの対応と個々の構造物生成関数をレンダラーと組み合わせることを考えたい。すなわち、風で枝が揺れるアニメーションを作成できるようにし、さらにレンダラーが画像生成時に必要な瞬間だけ構造物生成関数を呼び出して利用できるように構築したい。レンダラーが直接構造物生成関数を呼び出せるようになれば、ユーザーはレンダラーに対して、構造物を関数形式で記述したファイルを渡しレンダリング時にポリゴンが必要となるときにのみ、レンダラーが関数形式を自動的に妥当な精度でポリゴン形式に展開することができるようになる。従って、ポリゴンファイルのためにディスクスペースを消費する割合がさらに少なくなることが期待されるものである。実は、当システムの表現精度というパラメータは本来こういうことを行おうという目標を意識して、設定したものである。すなわち、最終的にレンダリングされたイメージにおいて、小さく表現されるべき物体は、表現精度（ポリゴン精度）を妥当なだけ粗くしてデータ量の削減とレンダリングスピードの向上を図ることが可能になるであろうと考えたわけで、遠い目標としては、構造物生成関数を内蔵した関数レンダラーを構築するという目標が存在することを述べておきたい。

６．参考文献

[１] Inakage, M., Approaches to Functional Based Modeling, SIGGRAPH 88 course note, pp.2-12

[２] Aono,M. and Kunii,T.L., Botanical Tree Image Generation, IEEE CG&A, May 1984, pp.10-34

[３] 金丸、千葉、斉藤，「ＣＧのための樹木の成長モデル」，第４回ＮＩＣＯＧＲＡＰＨ論文集，1988, pp.30-38

[４] 中嶋、福田、安居院，「景観表示のための樹木の生成手法」，第４回ＮＩＣＯＧＲＡＰＨ論文集，1988, pp.22-29

[５] J.Bloomental, Modeling the Mighty Maple, ACM.SIGGRAPH, 19, 3, 1985, pp.305 - 311