〔公　開〕

ＴＲ－Ｃ－００６７

ＣＧによる枝ぶり生成法

遠藤　陽

Akira　Endo

１９９１．９．１３

ＡＴＲ通信システム研究所

ＣＧによる樹木の枝ぶり生成法

ＡＴＲ通信システム研究所
知能処理研究室
佐藤　隆宣

龍谷大学理工学部
電子情報学科
T890132
遠藤　陽

目次

テーマ：　ＣＧによる枝ぶり生成法

実習期間：　1991・8/19～9/13

1　まえがき

距離を隔てた相手と、あたかも同じ場所（仮想空間）に存在するかのような環境を与える通信を、臨場感通信と呼んでいる。これは仮想空間に何人もの人が実際に存在しているかのように、話をしたり、物に触れたりする事ができる。この臨場感通信を利用して会議等を出来るようにと研究を進めているのが、ＡＴＲの通信システム研究所の知能処理研究室です。臨場感通信を実現するためにも、視線を検出し、視線に合わせて画像を変化できなければならない。頂点数の少ない画像では、滑らかに動かすことができるが、頂点数が多ければ多い程滑らかには動かなくなる。その頂点数の多い画像として、”樹木”がとりあげられている。私は、その”樹木”をより自然に、またはいろいろな種類に対応できるように研究を進めた。

2 本研究の必要性

景観図に使用される樹木のように、自然に近くなければならないのはもちろんとして、参考文献1に示されている通り以下の4つの条件を満たす必要がある。

条件１：極力共通のパラメータにより多種類の樹木に対応できること

条件２：同一樹木の変化に富んだ樹形を自動的に生成できること

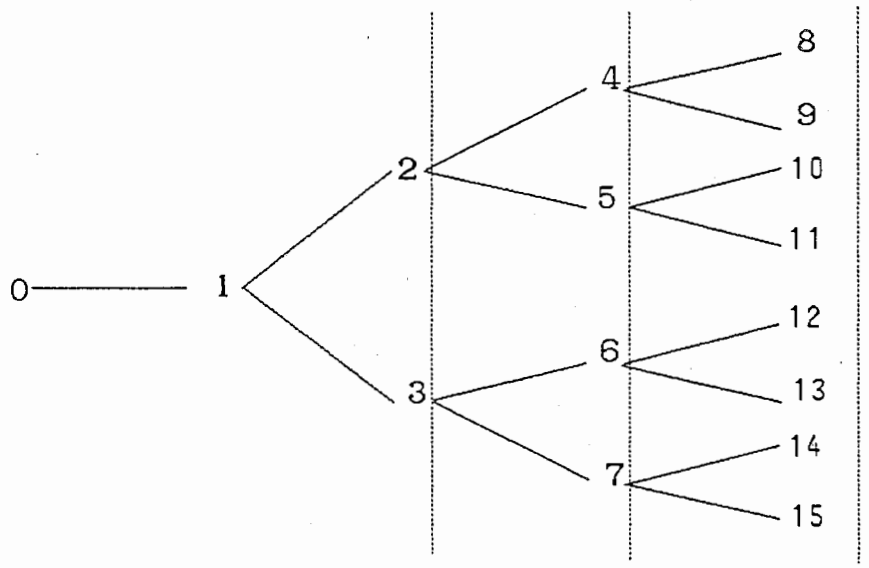
条件３：より自然な景観を生成させるため、各種の環境に対応できること。

条件４：経年変化にも対応できること。

以上のことは、臨場感通信に於ける階層化（頂点数）の変化にもつながりを持っている。そしてなによりも、視線の細かい動きに素早い対応をするためにも、頂点の座標の計算や表示などを素早く行う必要がある。それらの特に、”自然さ”“高速性”“階層化のしやすさ”を考慮した画像やプログラムを作成する必要がある。

3　従来技術の調査

球面体や立方体よりも頂点の多いものとして、”樹木”が使用されているが、今まで使用されていたプログラムでは、４つの条件のうちの２と４に対応しているだけで、あとの条件には余りふさわしくない。ｒａｎｄｏｍ関数で同一樹木の変化した樹形を形成することができ、世代数を入力することで経年変化にも対応していると考えられるが、一種類の樹木しかできず、全ての枝が同じ様にしか成長できないという欠点を持っている。また、”自然さ”“高速性”“階層化のしやすさ”の面でも、枝の生成の仕方（角度や長さ）も樹木らしさに欠け、頂点数の変化も世代数のみで決定するので、２の乗数の頂点数しか出せないという問題点もある。このプログラムでの分岐の方法と頂点番号の付けかたを図１に示す。



従来の分岐方法　　図　1

このプログラムの特徴として、先に全ての座標を指定して、後に一度に線を引くという方法を取っている。この方法のほうが一回一回線を引くよりも、”高速性”の面で優れている。しかしこれは、頂点番号をｉとすると、全ての面は［ｉ／２］につながっているという法則性をもっているからこそ簡単にできる。

頂点の座標の決定について述べると、角度と長さが重要な要素になっている。図２に、ｘ，ｙ，ｚ座標を表すパラメーターを示す。

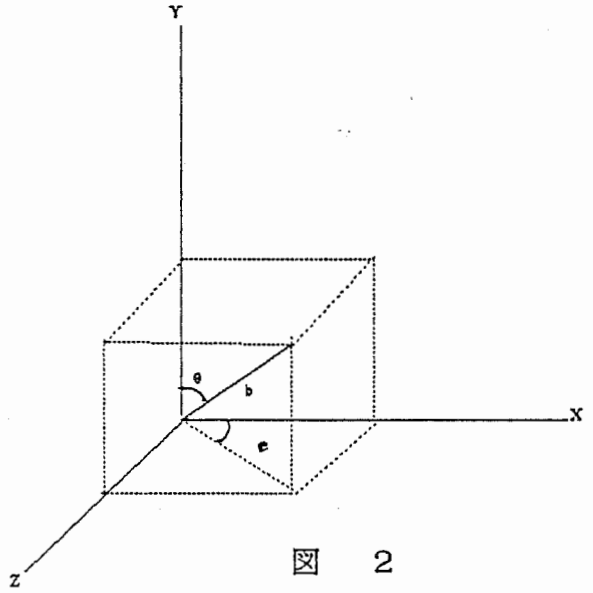


図　2

枝の角度の定義

図２により、 $x = b \sin\theta \cos\phi$、 $y = b \cos\theta$、 $z = b \sin\theta \cos\phi$ となる。 b, θ、 φについてもう少し詳しく述べると、 θは図３(a)のようにy軸に平行な直線を基準として、 $\theta = \theta_0 + \Delta\theta$ で与えられ、 θ_0は０で、 Δθは１５〔rad〕までの乱数で与えられる。尚、 θは $\pi/2 \geqq \theta \geqq -\pi/2$で、下向きには生成できないようになっている。 φは図３(b)のようにx軸に平行な直線を基準として、 $\phi = \phi_0 + \Delta\phi$ で与えられ、 ϕ_0は０で、 Δφは６０〔rad〕までの乱数を持っている。 bは $b_0 + \Delta b$ で与えられ、 b_0は２、 Δbは８までの乱数を持っている。また、頂点番号０－１間の枝を幹として区別して、bdで表し、長さを２０としている。尚、記号は私が使用した記号を利用している。

このプログラムを基として、研究を進めて行く。

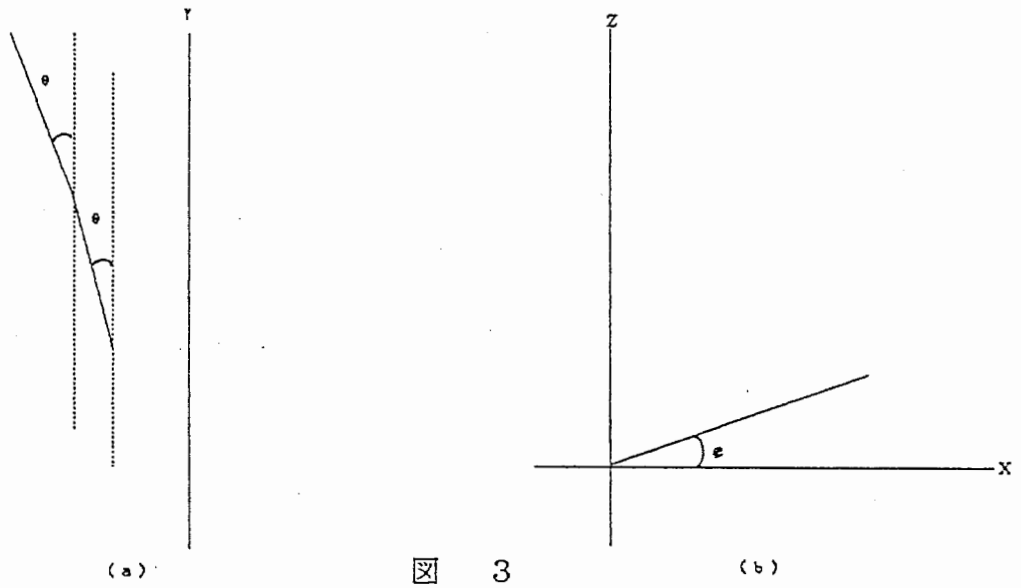


図　３

従来の角度の定義の仕方

4　本研究の特徴

4・1　栄養配合を考慮した枝生成アルゴリズム

4・1-1　”階層化のしやすさ”を考慮したプログラム（ファイル：report1.c）

階層化をし易くするためには、頂点数をいろいろと変化できなければならない。そのために各々の枝の成長の仕方に、変化を与えてやる。参考文献2に従い、栄養分の分配を行う。その栄養分の分配比率fを変化させることにより、多種類の”樹木”を生成して、頂点数にバリエーションを持たせる。その栄養分の分配法を、f＝1/3として、図4に示す。

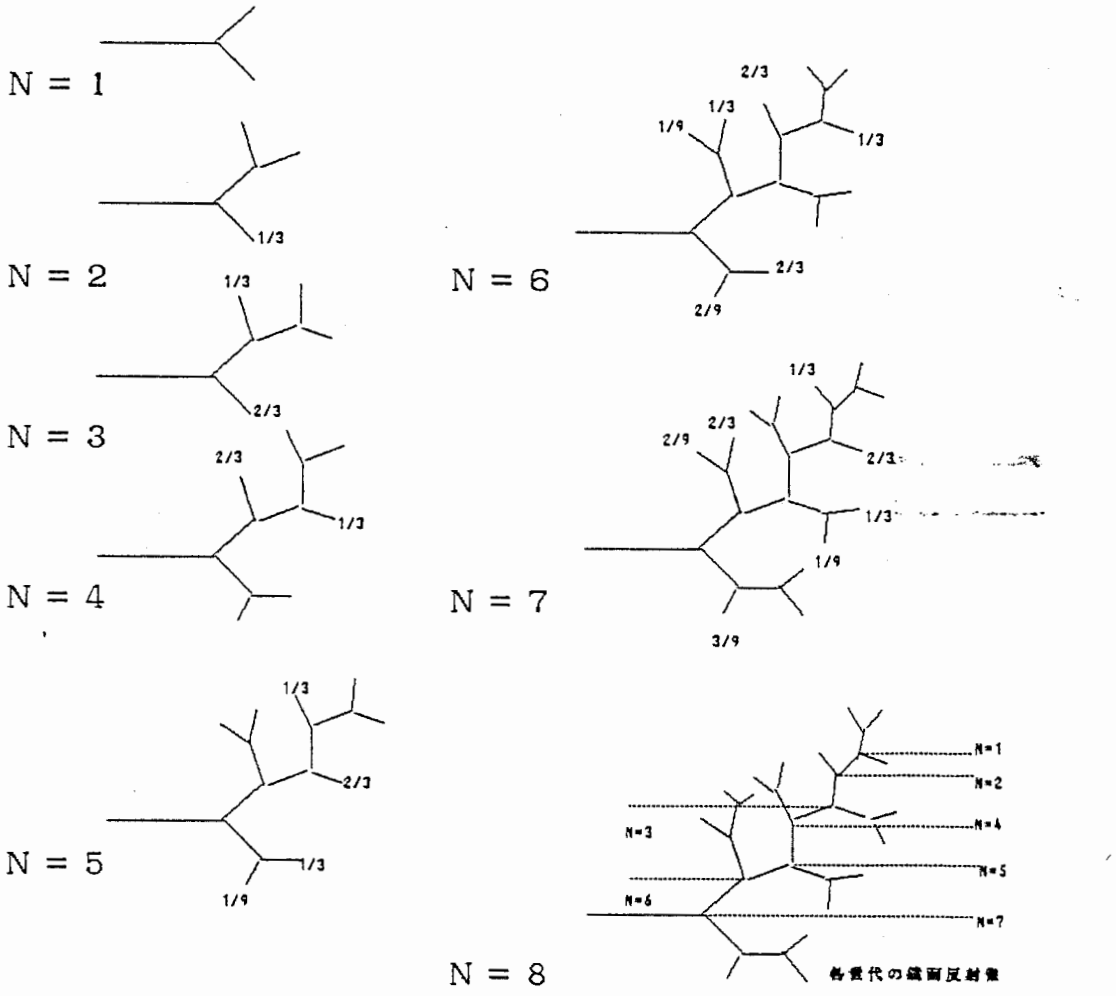


図　4

fの異分分配法

図４のように受け取った養分の合計が、１を越えるものだけを分岐させる。二又分枝のうちの一本を、親枝の継続と考え、次の養分分配で一本は親枝と同じ量を、もう一本が親の量のｆ（０≦ｆ≦１）倍を受け取る。図４の、Ｎ＝４のときのように、親点が受け取っていた養分が1/3なら、子供はＮ＝５のとき、1/3と（1/3）の２乗を受け取る。さらに枝分かれして中心から離れる毎に、養分量は少なくなる。文献に従い、頂点番号は発生順に付け、一回の養分分配で同時に発生した複数の枝を同世代とする。図５に、西洋数字を頂点番号、ローマ数字を世代番号で表し、ｆ＝1/3とする例を示す。

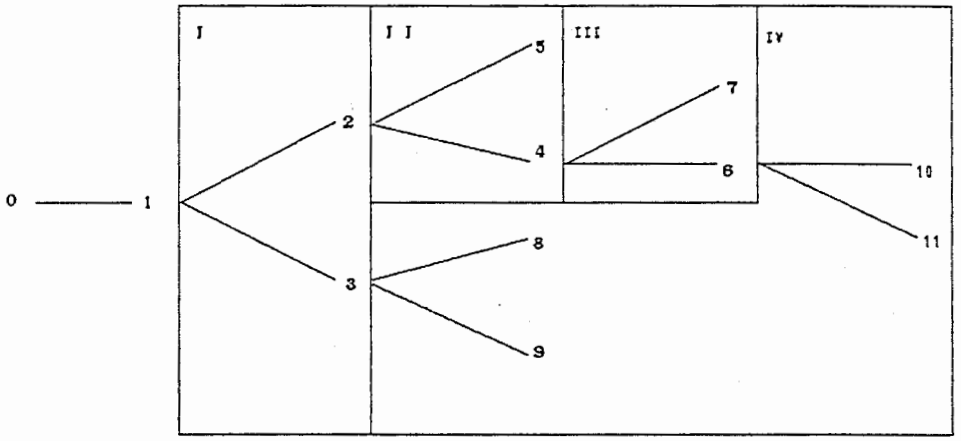


図　５

養分分配を考えた生成方

以上の理論でプログラムを作成するとしての必要事項について述べる。まず、分岐した枝に”２”と”０”を与えてやる。”１”ではなく”２”にしたのは、０を与えられた点が何回かの養分分配で、養分量の合計が”１”になり、１を与えられた点と間違うからである。分岐したときに”２”を与えられるものは、幹の継続と考えるので、以後ここでは、”主幹”と呼ぶ。頂点番号の１番から順に、そのときの頂点番号の最大の番号までを探索して、分岐可能で主幹でないもの（主幹は、初めから１を越えているので、養分分配はいらない）に養分

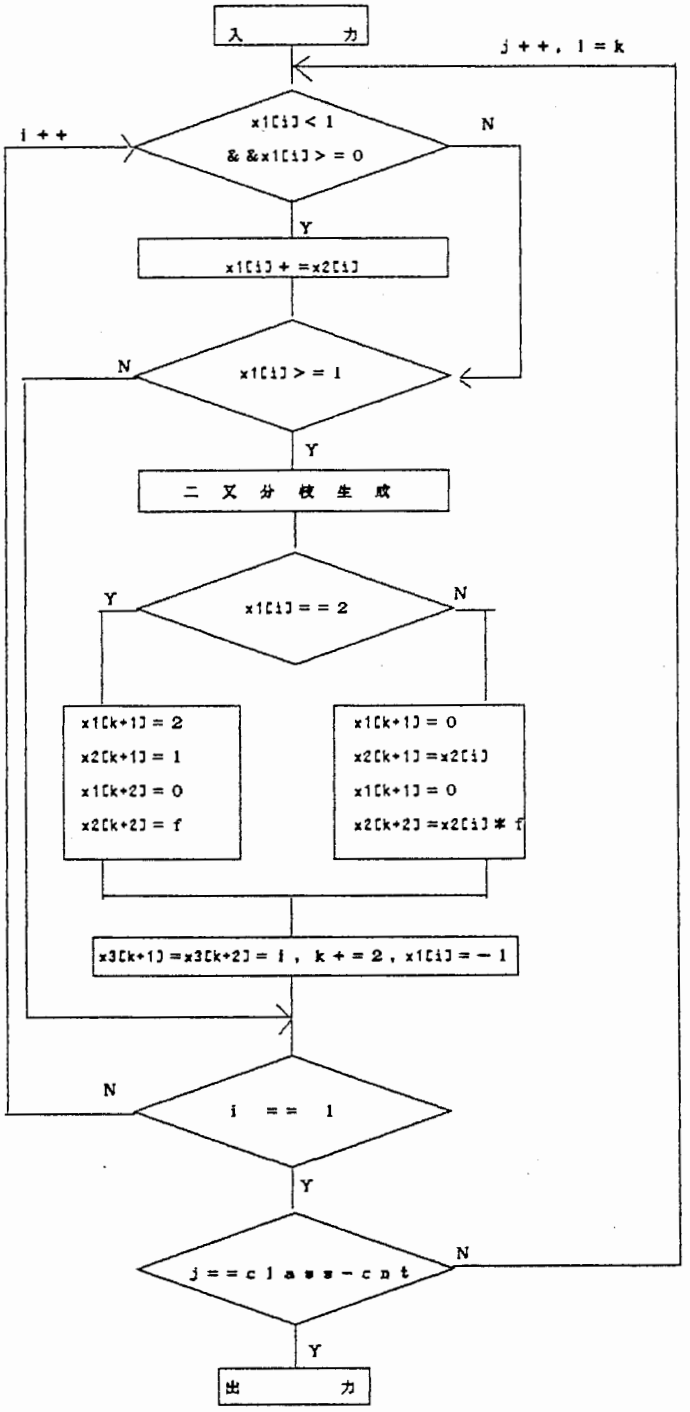
分配を行い、主幹を含めた分岐可能なもの（ここでは端点と呼ぶ）で、1を越えているものを分岐させてやる。分岐したなら、発生順に頂点番号を付けてやる。その際、2本のうち1本は親枝の継続と考えるので、親点の養分分配量と同じ分配量を、もう一方は、親の分配量のf倍を次回から受け取る。

これらをプログラムの記号でいうと、現在持っている養分量をx1〔i〕（iは頂点番号）にいれ、1回の養分分配量をx2〔i〕にいれる。主幹はx1〔i〕として2を持ち、分岐の終わった点（分岐可能でない点）には-1をいれる。そして、線を引くときのために、その点の親の点の番号をx3〔i〕に記憶させておく。

世代数の入力値をclass-cntとし、角度、長さについては以前の方法でプログラムを作成した。このプログラムのフローチャートを図6に示す。尚、初期値としてx1〔1〕=2，x2〔i〕=1，x3〔i〕=0とし、変数をk=l=i=j=1とする。

4・1-2 ”高速性”を考慮したプログラム（ﾌｧｲﾙ: report2.c）

”高速性”と言うほど大げさなものではないが、私が作成した箇所だけで、処理等の短縮がはかれないかと考えた。report1.cのプログラムでは、分岐し終わった点までも調べている。これは無駄だとなので、新しい頂点番号の付け方を考える。図7のように分岐し終わった点の番号を推移させて、端点だけを調べる方法をとる。この方法でいくと、分岐する可能性の無い点は、調べなくて良くなり、座標は、分岐時にreport1.cと同じ様な頂点番号の付け方で、憶えさせておく。（図7では、○の中の数字）



主幹であるもの（x1[i]＞1）
分岐する可能性のないもの（x1[i]＜0）はN

微分分配（主幹は除く）

微分が1以上あれば分岐

主幹ならY，そうでないならN

Y：k＋1に主幹の値2を与え、k＋2に
主幹から分岐した枝の値を与える
N：k＋1は親枝の継続とみなし、同じ値を与
え、k＋2は主幹からどれだけ離れたかに
応じた値をあてる

頂点の記憶と、分岐終了した点に－1を与える

図　　6　report1.cのフローチャート

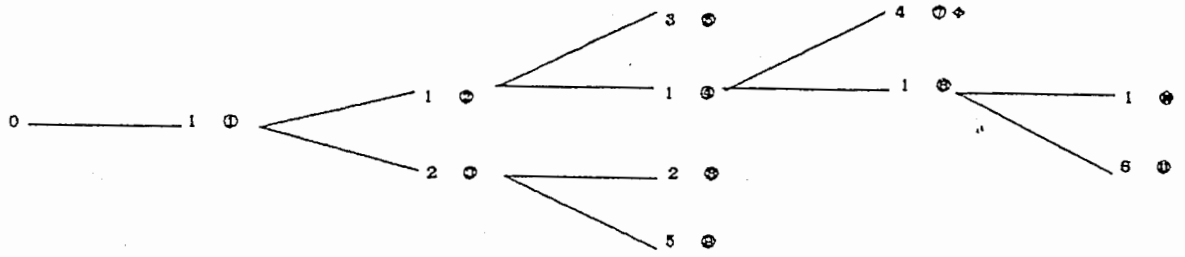


図　7

プログラムでは分岐させるときに、新しくできる点は親の点の座標を基準とするので、端点が今どこの頂点にいるのかを、憶えさせておく必要があり、そのために、x4[i]という配列名を使う。iは端点のみの番号で、iがどの頂点にいるのかをx4[i]が、分岐して次の端点に移るまで記憶しておく。以上のフローチャート（図6の”二又分枝生成”以降の一部のみ）を図8に示す。なお、m=1を変数の初期値としてつけ加える。

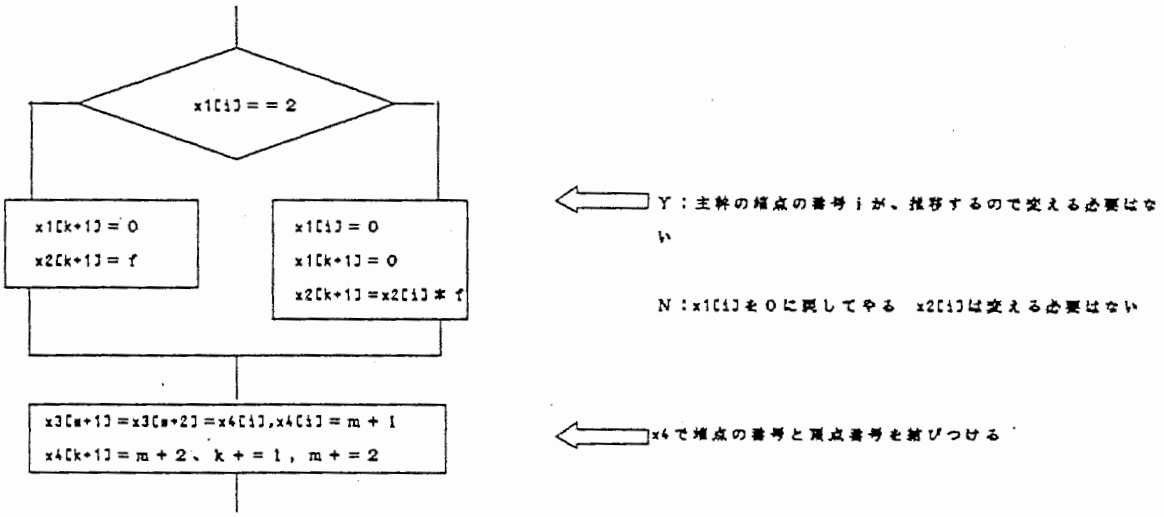


図　8

report2.cのフローチャート

4・1-3 ”自然さ”を考慮したプログラム（ﾌｧｲﾙ: report3.c）

上記2つのプログラムでは、角度や長さに手を加えていないので、樹木らしくない。これを樹木らしく見せるために、θ_0、ϕ_0、$\Delta\theta$、$\Delta\phi$、b_0、Δb,bdをパラメータとして入力できるようにする。尚、角度は、度数入力に変更している。改良点を、θ、ϕ、bに分けて説明する。

θ） θ：$-\pi/2 \leqq \theta \leqq \pi/2$でないと決定のやり直しがあったが、それぞれの限界を越えると、限界付近の値を持つようにした。

θ_0：今までは0に固定されていたが、最低限開いて欲しい角度として入力する。

$\Delta\theta$：θ_0からどれだけの（ランダムな）角度開いているかをあらわす。今までは、$0 \leqq \Delta\theta \leqq 15$[rad]でしたのを、樹木に応じてθの最大θ-max（最小の角度と最大の角度の差：$0 \leqq \Delta\theta \leqq \theta$-max）を入力する。

（図9参照）

ϕ） ϕ：x軸に平行な直線を基準にしていたが、枝らしく見せるために、親点の角度を基準として、正負両方向に開く。

例外として、頂点番号1からでている2本の枝に関しては、同じ方向に生成するとバランスが悪いので、180度程度離す。

ϕ_0：今までは0に固定されていたが、入力という形をとっている。主に0を使う。

Δφ： 0≦Δφ≦15〔rad〕という何回転でもできる値を取っていたが、枝と枝が180度ずれると枝らしくなくなるので、今回の入力値としては、基準を親の点の角度に変えたところから、90度前後が適当。これも入力としては、φ-max（0≦Δφ≦φ¥max）を入力する。

（図10(a)(b)参照）

b） bd： 幹の長さの入力（今までは20に固定）。

bø： bの最低限あって欲しい長さ。今までは2だったが、樹木によって入力ずる。

Δb： bøに足すランダムな長さを表す。0≦Δb≦8だったが、Δbの最大b-max（0≦Δb≦b-max）を入力する。

（図10(c)参照）

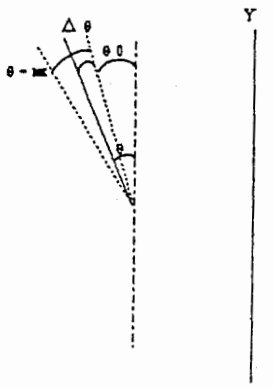


図 9

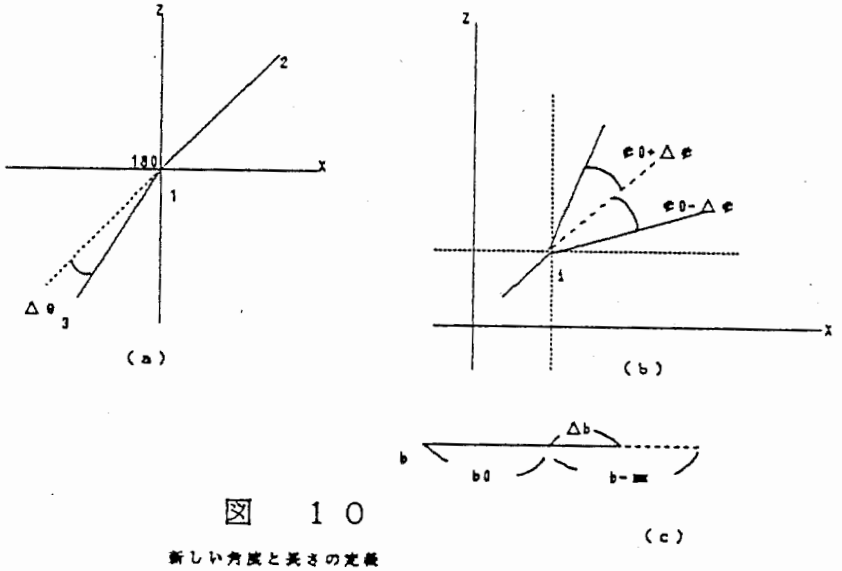


図 10

新しい角度と長さの定義

について、入力値をthrとして以下に示す。

thq)　全ての枝にたわみを付けるのではなく、図11に示すように主幹から1、2回目に分岐した枝につながる枝全部に、おもみをつける。つまり、頂点番号の3、5の頂点に、図11でいうと、○印に重みを付ける。2本のうち1本は親枝の継続と考えるので、a,b,cはそれぞれ同じ値を持つ。aに(1+thq),bに(1+thq+thq),cに(1+3*thq)、さらに主幹から離れる毎にthqが加算される。ここでは(1+th[i])として表す。(0≦thq≦1)ぐらいが適当で、0にするといままのθと同じになる。

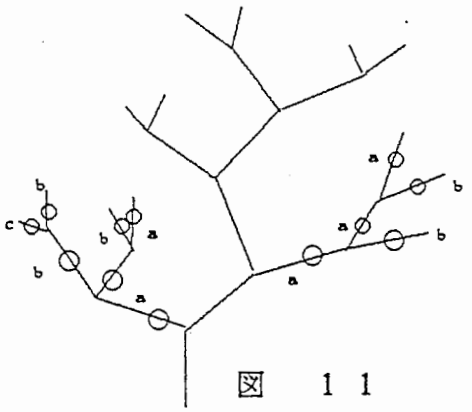


図　11

枝のたわみを付ける位置

thr)　杉のような真っ直ぐに延びる主幹を作るために、主幹のθ成分を狭くする。thrは(0≦thr≦1)の値をとり、0を入力するとθa、θ-maxの値と関係なく0度に設定される。0に近付くほど、主幹の真っ直ぐな杉に近付く。逆に、1に近付くほど、桜等の主幹の真っ直ぐに成長しない樹木に近付く。

以上のことを、図１２にフローチャートで表す。（report1.cの”二又分枝生成”の中に相当する。）

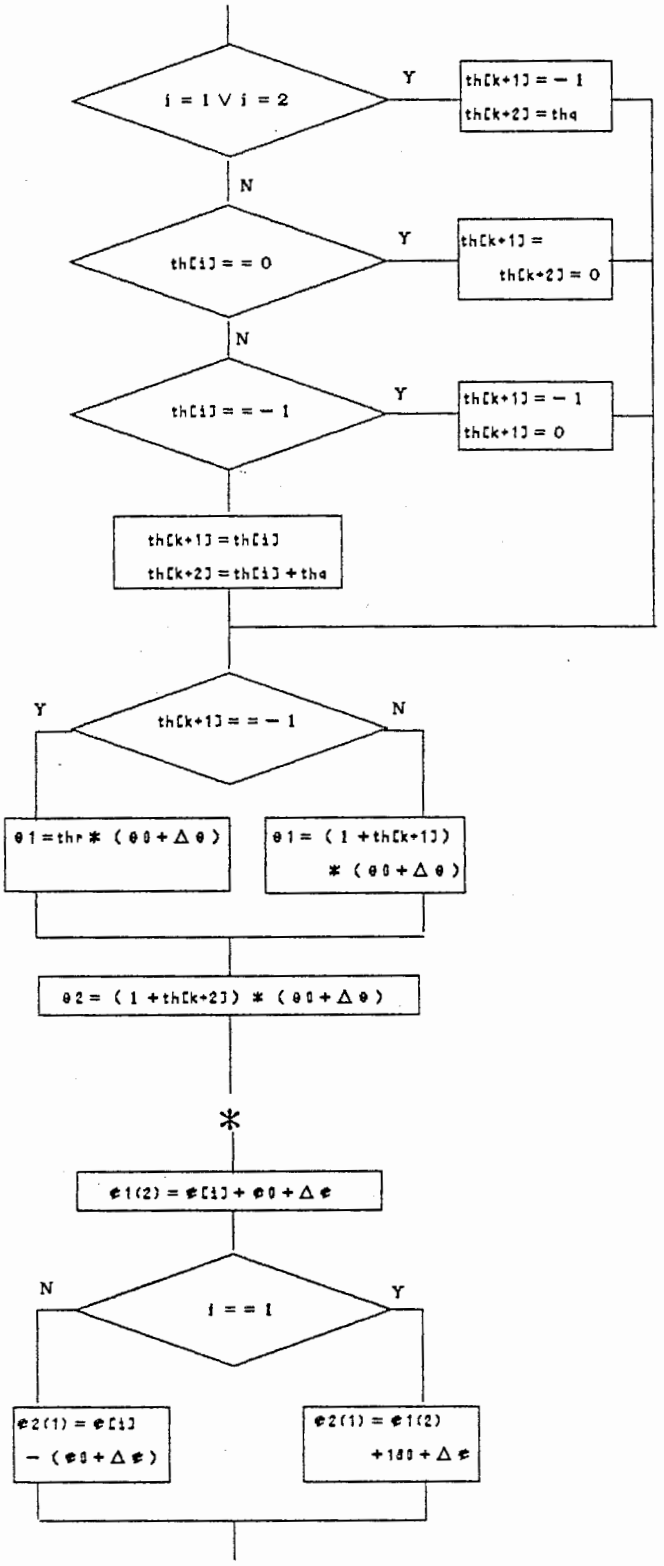
以後、report3.cのプログラムを使用する。

4・2　樹木の種類とパラメータの関係

まず、図１３を使って、実際使っていく上でのパラメータの使い方について説明する。

class-cnt	世代数　1～10程度
set-rnd	乱数の元になる数
f	養分分配比　0～1
θ₀	最低限開けたい角度　0～90程度まで
φ₀	最低限開けたい角度　普通0を使用
θ-max	大きくすると枝の生成角度に幅がでる。0～30程度
φ-max	90程度
thq	枝のたわみを表し、0は重みが加わらない。0～1程度
thr	主幹の伸び方で、真っ直ぐは0。0～1程度
bd	（最下層）の幹の長さ。　20程度
b₀	枝の最低限生成して欲しい長さ。1～10程度
b-max	大きくすれば、枝の長さに違いを持たすことがでる。1～5程度

表 1



頂点番号３、５を見つけ、重みを加え、
２、４は主幹なので、目印に”－１”をつける

関係ない枝を見つける

主幹をみつけ”－１”をつける

３、５につながる枝なので、それなりの値を
与える

主幹かどうか判別

Ｙ：主幹としてθにthrをかける
Ｎ：関係のない枝はth[k+1]が０なので、
（１＋th[k+1]）をかける

上のＮと同じように（１＋th[k+2]）
をかける

範囲外を見つけ、θ＞＋PI→＋PI－△θ
θ＜－＋PI→－PI＋△θ

１ｏｒ２かはランダムにあたえる
１なら２、２なら１を下で与える

頂点番号１から出る枝を
180度程度開ける

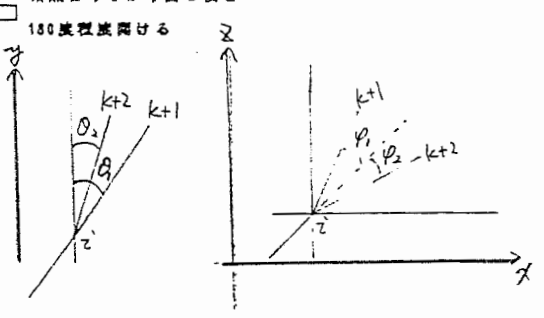


図　　１２　　report3.cのフローチャート

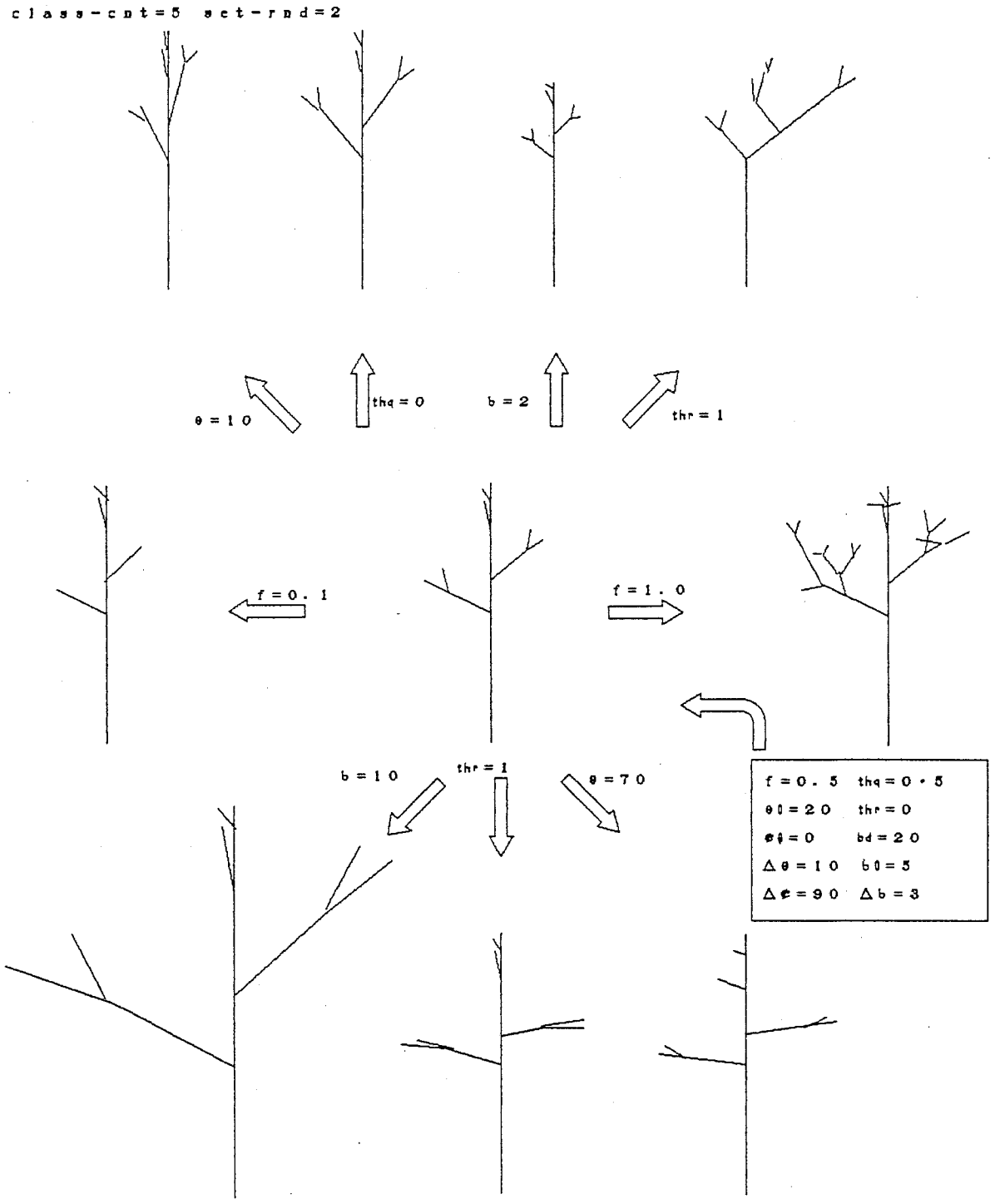


図　13

パラメータによる樹木の変化

次に、参考文献に例としてあげられている４つの樹木（a.ケヤキ、b.ソメイヨシノ、c.イチョウ、d.プラタナス）とそのほか４つの樹木について、自分なりのパラメータの入力値を探したので、掲示する。

	a	b	c	d	e	f	g	h
class-cnt	10	10	13	10	10	10	10	10
set-rnd	3	3	4	3	3	2	3	2
f	0.95	0.99	0.45	0.6	0.9	1	0.78	0.2
θ_0	10	20	25	30	20	40	40	10
ϕ_0	0	0	0	0	0	0	0	0
ϕ-max	40	20	25	30	10	40	10	20
ϕ-max	90	90	90	90	90	90	90	90
thq	0.2	0.5	0.5	0.6	0.1	0.2	0.1	0.4
thr	0.7	0.6	0.1	0.1	0.9	0.6	0.3	0.1
bd	20	15	10	15	2	3	30	30
b_0	5	5	6	6	10	3	2	5
b-max	3	3	2	2	5	6	3	2

表　2

以上の生成結果のイラストを図１４（a）～（h）に示す。

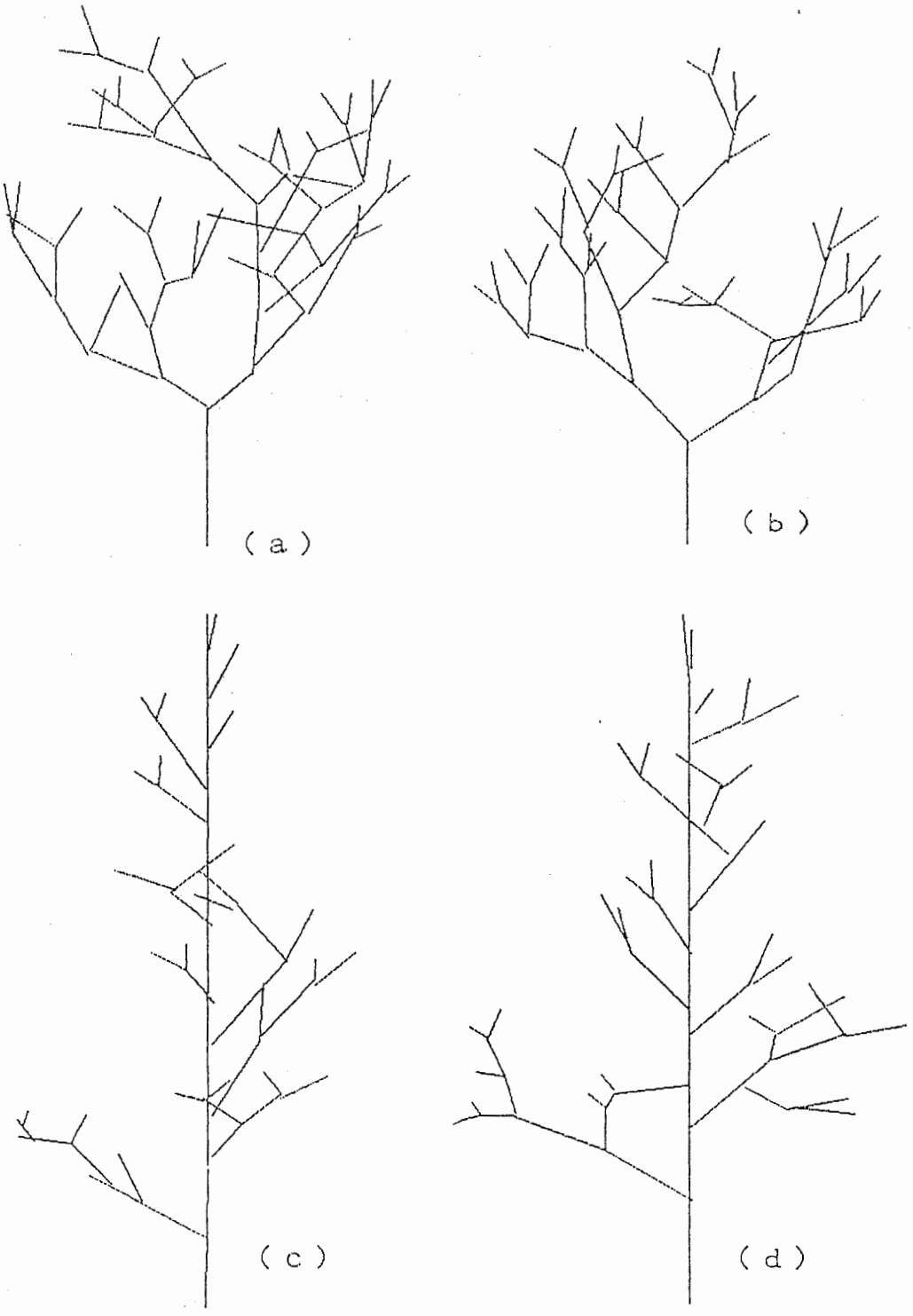
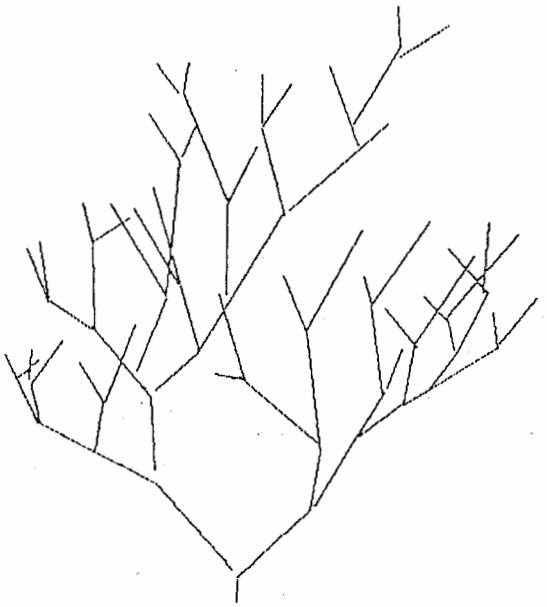
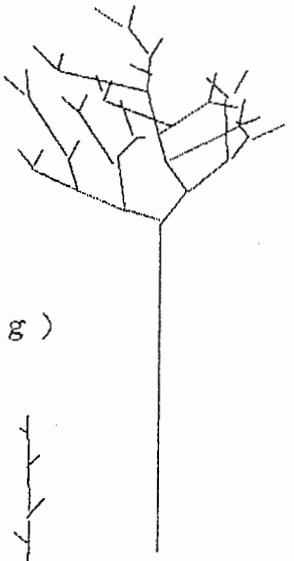


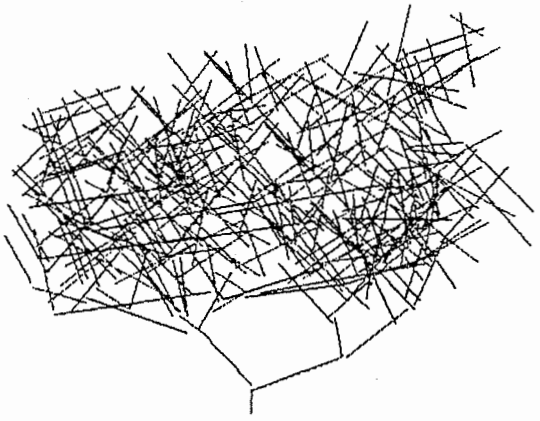
図 1 4



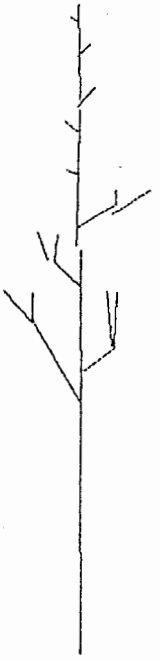
（ｅ）



（ｇ）



（ｆ）



（ｈ）

図　１４

4・3　幹と葉の接続

report3.cのプログラムに、幹と葉を付けてもらった。葉を接続してもらうために、端点とその親の番号を計算するプログラムをつけ加た。それについて少し説明する。（後に付けたreport3.cのプログラムの"leaf make"参照）まず、trunkという配列に、端点でない頂点の番号をいれる。その方法としては、端点でない点はx3[i]に番号があるもので、x3[i]のi=2からは2つずつ同じ番号が並んでいるので、偶数だけを見てやればよく(i=1の親は0なのでみる必要がない)、それを番号順に並び変えて、trunkにいれてやる。図5でいうと、1～11までの親点の番号(0,1,1,2,2,4,4,3,3,6,6)が、単一化され、並び替えられて(1,2,3,4,6)としてtrunkの中にはいる。次にleafという配列の中に、端点をいれてやる。まずtrunkの中の最大の数字（図5では6）までで、trunkにない数字（図5では5）を探す。そしてtrunkの中の最大の数字より大きい数（図5では7,8,9,10,11）も端点なので、その後にいれてやる。次にその親点を探すが、普通に探したのでは、同じ数字（親点1個に対して、子供の点は2つなので）が2個でてくる可能性があるので、leafの中の順番(leafの中はソーティングされている)が一つ後の頂点と親が同じならleafの中に付け加えないことにしている。図5でいうと、今leafの中には(5,7,8,9,10,11)がはいっていて、8と9,10と11がそれぞれ同じ親を持つので、小さい方の数字は調べない。これで端点とその親の頂点は、leafの中に記憶される。

葉と幹を接続するプログラムについて少し述べる。幹や枝は、円（aとして半径を入力）に内接するn角形（cとしてn/2を入力）の柱を使い、柱を何分割（dとして入力）かして、縮小率（bとして入力）に従って細くする。葉は枝の分割数（d）に伴って付ける。形は

円（ｅとして入力）に内接する六角形の柱を、ある角度（ｇとして入力）で切断したその面を葉として用いる。以上のことがパラメータ入力できるようになっているので、少し説明する。

ａ　頂点数０の底面となるｎ角柱の内接する円の半径。

ｂ　親点と子供の点の縮小率。

ｃ　幹や枝をｎ角柱にするとき、ｎ/２を入力。

ｄ　枝の分割数の入力。

ｅ　葉の元となる六角柱の内接する円の半径。

ｆ　葉をどの点に付けるか。（指定済みなので不要）

ｇ　葉の形をしていするために、切断角度をしていする。

4・4　生成結果の評価

4・3でとりあげた８つの樹木のモデルに、幹と葉を付け、木の頂点数、葉の頂点数（葉１まいで、６個の頂点と１個の枝との接続点を持つ）とその合計、そして１枚の画像を出すのにかかる時間を示す。
（a'はaの世代数を13にしている）

	tree-point	leaf-point	total	time(msec)
a	3906	2940	6846	109.960937
a'	10626	7980	18606	275.000
b	3906	2940	6846	110.058594
c	2982	2352	5334	83.007813
d	2730	2142	4872	91.992188
e	3906	2940	6846	110.058594
f	42966	32214	75180	609918.457031
g	3486	2688	6174	110.058594
h	1134	924	2058	34.082031

表 3

5 考察と今後の課題

上述の結果から、頂点数の数が増えれば増えるほど時間がかかっていることがわかる。fの結果を無視すれば、1msecあたり50-70ぐらいの頂点数を計算していることになる。この結果からも、値としてはいい結果が得られたと思う。頂点数の方もかなりの変化が出来るようになった。プログラムのスピード・アップ(report2.c)は、report1.cと比べても余りかわりなかった。プログラムでの"高速化"には、あまり効果はないようだ。"自然さ""階層化のしやすさ"、そして4つの条件についてはほぼ満たせた。ただ、葉の数が変化しないので、季節感(条件3)が出せなかった。

今まで進めてきて残念なことは、葉を付ける数が少ないのと、leafで指定している頂点が少ないこと。先ほど述べたleafの中を改良すれば、解消できると思う。私の案としては、leafの中をもう一度ソーティングして、その親点を同じ方法、または別の方法で探してくるというのと、leafの中に同じ番号が入っていいのなら、親点を探すときから、x3[i]に好きなだけいれるだけで出来る、という方法である。葉を付ける枝が指定できたら、頂点数がもっと変化しやすくなると思う他の改良点として、枝の重さを計算して、たわみを付けたり、枝の長さを縮小率等で制御したらいいと思う。後、枝の重なりをなくしたり、日光照射量を考慮できたのなら、すばらしいものが出来上がると思う。そして”高速性”に関しては、まだまだ研究する必要があると思う。

謝辞

この度の実習の機会を頂きました寺島社長に、また実習に関わりお世話になりました岸野室長、伴野さん、鉄谷さん、境野さんに、樹木に幹と葉をつけて頂きました佐川さん、そしてＡＴＲのみなさんに心から感謝いたします。

参考文献

1 安居院、福田、中嶋：景観表示のための樹木の生成手法、情報処理学会論文誌 Vol.32 No.5 pp. 618-625(1991)

2 Honda,H.,Tomlison,P.B.andFisher,J.B.:Computer simulation of Branch Interaction and Regulation by Unequal Flow Rates in Botanical tree,Am.J.Bot.,Vol.68, No.4, pp. 569-585(1981)

```
/*****************************************/
/*   make tree                           */
/*****************************************/
int class_cnt;
int setrnd;
float f,thq,thr;
int th_max,del_max,b_max;
int bd,th0,del0,b0;
int x3[NN],leaf[NN],trunk[NN],l,x4[NN];
int number_of_treepolys;    /* tree polygon count */
int number_of_polys;
int number_of_leafpolys;
POLYGON *treelinepolygon;       /* pointer to polygon list */
POLY    *hexpolygon;
POLYDAT *leafpoly;
/*****  1991/09/04  *****/
char   namein[20], getbuf[100];
int    n, cut, lx, dt, st;
int    treecount = 0, leafcount = 0;
float  rndata, ptheta, dtheta, step, ltheta;
float  rt, rw, rb, tlb, tlx, tly, tlz, al, leafr;
float  rad = PI/180.0;

float p0[3],p1[3],p2[3],p3[3];
float c0[3],c1[3],c2[3],c3[3];
float n0[3],n1[3],n2[3],n3[3];

make_tree()
{
  POLYGON *p;
  POLY    *pt , *Bpt , *Wbpt , *Wtpt;
  POLYDAT *leafpt;

  float bx,theta, delta;
  int   temp,cnt,lcnt,bcnt;

  int vertex_i,hexdat_i;
  int i, j, k=1,r,g;
  float b[3],t[3];
  float x1[NN] ,x2[NN],del[NN],th[NN],rn[NN],bdt[NN],thdt[NN];
  float v1[3],v2[3];
  dtheta = PI / n;

  number_of_treepolys =  pow(2, class_cnt+1) * 3;
  p = treelinepolygon = (POLYGON *)malloc(number_of_treepolys * sizeof(POLYGON))
;
/*****  1991/09/04  *****/
  number_of_polys = 2*n*cut*(pow(2,class_cnt)-1)
    + 2*n*(2*(pow(2,class_cnt-1)-1));
  pt = hexpolygon = (POLY *)malloc(number_of_polys * sizeof(POLY));

  Bpt = pt;
  number_of_leafpolys = cut* (pow(2,class_cnt)-1);
  leafpt = leafpoly = (POLYDAT *)malloc(number_of_leafpolys*sizeof(POLYDAT));
/************************/
  srandom(setrnd);
  p->vertex[0].x = p->vertex[1].x = 0 ;
  p->vertex[0].y = 0 ;
  p->vertex[0].z = p->vertex[1].z = 0 ;
  p->vertex[1].y = bd ;
  x1[1] = 2;x2[1] = 1;x3[1] = 0,del[1] = 0;
  x4[1] = temp = 1;
  for(j = 1; j <= class_cnt; ++j){
    for( i = 1; i <= l ;i++){
      if(x1[i] < 1 && x1[i] >= 0)
        x1[i] += x2[i];
```

＊x3[i]は親の点の番号

＊x1[i]は現在持っている着分量
(主幹は2，分岐の終わった点は-1を持
＊x2[i]は1回の着分配で受けとる量

入力した世代数まで生成

主幹であるもの(x1[i]>1)
分岐の可能性のないもの(x1[i]<0)　以外は着分分配を行う

```
    if(x1[i] >= 1){
      switch(i) {
      case 1:th[2] = -1;th[3] = thq;
        break;
      case 2:th[4] = -1;th[5] = thq;
        break;
      default:if(th[i] > 0){
        th[k+1] = th[i]; th[k+2] = th[i] + thq;
      } else {
        th[k+1] = th[i]; th[k+2] = 0;
      }
      }

      if(th[k+1] == -1)
        t[1] = thr * (th0 + random()%th_max) * PI /180;
      else
        t[1] = (1 + th[k+1]) * (th0 + random()%th_max) * PI /180;
      t[2] = (1 + th[k+2]) * (th0 + random()%th_max) * PI /180;

      for(g=1;g<=2;g++){
        if(t[g] > PI /2)
          t[g] = PI /2 - (random()%th_max) * PI /180;
        else  if(t[g] < - PI /2)
          t[g] = - PI /2 + (random()%th_max) * PI /180;
      }

      if(i != 1){
        r = 1 + random()%2;
        del[k+r] = del[i] + (del0 + random()%del_max) * PI /180;
        if(r==1)
          del[k+2] = del[i] - (del0 + random()%del_max) * PI/180;
        else
          del[k+1] = del[i] - (del0 + random()%del_max) * PI/180;
      } else {
        del[2] = (del0 + random()%del_max) * PI /180;
        del[3] = del[2] +(180 + random()%del_max) * PI /180;
      }

      for(g=1;g<=2;g++){
        b[g] = b0  + random()%b_max;
        thdt[k+g] = t[g]; bdt[k+g] = b[g];
        p->vertex[k+g].x =p->vertex[i].x +b[g]*sin(t[g])*cos(del[k+g]);
        p->vertex[k+g].z =p->vertex[i].z +b[g]*sin(t[g])*sin(del[k+g]);
        p->vertex[k+g].y =p->vertex[i].y +b[g]*cos(t[g]);
      }
      if(x1[i] == 2){
        x1[k+1] = 2;x2[k+1] = 1;x1[k+2] = 0;x2[k+2] = f;
      } else {
        x1[k+1] = 0;x2[k+1] = x2[i];x1[k+2] = 0;x2[k+2] = x2[i] * f;
      }
      x1[i] = -1;x3[k+1] = x3[k+2] = i;
      k += 2;
    }
  }
  l = k;
}
/*****  tree data make  ***  1991/09/04  *****/
  for(i=2,x4[1]=0; i <= l; i++){
    cnt = x3[i];
    for(temp=1;cnt != 1;){
      cnt = x3[cnt];
      temp++;
    }
    x4[i] = temp;
  }
```

頂点番号3,5をみつけ たわみをつける
〃 2,4は主幹なので目印に"-1"をつける
主幹でも たわみをつけるものでもないもの.
{主幹に"-1"をつけ
3,5につながる枝にそれなりの値を与える

主幹かどうか
主幹なら それをかける.

{たわみをつける
関係のない枝は
(それぞれが0)

$\theta > \frac{\pi}{2}$ なら $\frac{\pi}{2}$ 弱の値を与える
$\theta < \frac{\pi}{2}$ なら $-\frac{\pi}{2}$ 弱の値を与える

for2をランダムに与える

頂点番号7から出る枝を180度程度開ける

座標を決定する

主幹かどうか判別.
(k+1を主幹, k+2を主幹から分岐した枝とする
{主幹からどれだけ離れているかによって値を与える

```
/*****  leaf make  *****/
  k=0, r=0;
  for(j=1;j<=l;++j){
    for(i=1;i<=l/2;++i){
      if(j == x3[2*i]){
        trunk[k+1] = j;
        k++;
        break;
      }
    }
  }

  for(i=1;;++i){
    if(trunk[i-r] != i){
      leaf[r+1] =i;
      r++;
    }
    if(i-r == k) break;
  }

  for(i=1;;++i){
    leaf[r+1] = trunk[k]+i;
    r++;
    if(trunk[k]+i == l) break;
  }
  for(i=1;;++i){
    if(x3[leaf[i]] != x3[leaf[i+1]]){
      leaf[r+1] = x3[leaf[i]];
      r++;
    }
    if(leaf[i] == l) break;
  }
/*****  leaf data make  *****/
  for(i=1;i<r;++i){
    cnt = leaf[i];  j = x3[cnt];
    tlb = bdt[cnt]/(float)cut;
```

＊l：頂点番号の最大

x3[i] は親番号なので それに含まれらい番号は端点。x3[i]の中の番号は i=2 のときから、2つずつ同じ番号が並んでいる (x3[1]=0 で0には葉をつけない)ので偶数番号だけを見てやり x3[i]に含まれる番号を0を除いて 単一化し、ソーティングする。

＊trunk[k]：端点でない(分岐している)点の最大

1～trunk[k]までで端点なもの (trunkの中では数字がとんでいるはず)を leafに入れる。

trunk[k] より大きい番号の点は端点なので順番に leafに追加する。

leafの中の番号の親点を見つける。
leafは小さい順に並んでいるので、同じ親をもつ点はとなり合っている。その大きい方だけをleafと 同じ親をもつ点がないものをleafに追加する。

```
for(i=1.rn[0] = rndata; i <= l; i++){
  rn[i] = rndata;
  for(j=0;j <= x4[i];j++)
    rn[i] = rn[i]*step;
}
for(i=1; i <= l; i++){
  j = x3[i];
```