Nous donnons encore, ci-dessous, un algorithme pour calculer les inconnues t, f_1, f_2, f_3. L'inconnue positive t satisfait à une équation biquadratique

$$0 = [f^2, f^2] = [t^2a^2 - d^2, t^2a^2 - d^2] =$$

$$= t^4[a^2, a^2] - 2t^2.[a^2, d^2] + [d^2, d^2].$$

L'existence d'une solution t, implique que le discriminant

$$D = [a^2, d^2]^2 - [a^2, a^2] \cdot [d^2, d^2]$$

satisfait à l'inéquation $D \geq 0$.

Si $D = 0$ on a

$$t^2 = \frac{[a^2, d^2]}{[a^2, a^2]} = \sqrt{\frac{[d^2, d^2]}{[a^2, a^2]}},$$

$t^4[a^2, a^2] = [d^2, d^2]$. Ceci veut dire que l'aire du triangle $\{t\,a_1, t\,a_2, t\,a_3\}$ ne sera pas diminuée après une projection orthogonale. On a $f_1 = f_2 = f_3 = 0$ et la projection P sera orthogonale.

Si $D > 0$, il existera deux solutions différentes

$$t_1^2 = \frac{[a^2, d^2] + \sqrt{D}}{[a^2, a^2]}, \quad t_2^2 = \frac{[a^2, d^2] - \sqrt{D}}{[a^2, a^2]}.$$

Les conditions

$$t_1^2 > 0, \quad \frac{[d^2, d^2]}{[a^2, a^2]} > 0$$

impliquent $t_2^2 > 0$, $[a^2, d^2] > 0$, d'où on a $t_1^2 > t_2^2$. On obtient ensuite

$$\frac{[d^2, d^2]}{[a^2, a^2] \cdot t_2^2} > t_2^2,$$

$[d^2, d^2] > [t_2^2 \cdot a^2, t_2^2 \cdot a^2]$. L'aire du triangle $\{t\,a_1, t\,a_2, t\,a_3\}$ sera augmentée après une projection orthogonale. La solution t_2^2 est fausse (cf. fig.6).

Par le programme suivant on considére cas plus gén'erale qui selui du théorème de Pohlke-Schwarz. Nous supposons qu'il existe trois segments non-coplanaires au plan, cést à dire qu'un des nombres complexes z_1, z_2, z_3 est certainement non-nul. Le programme va chercher le nombre σ_i, $i = 1, 2, 3$ le plus commode.

En modelant, nous comparons souvent deux nombres réels: x et 0. On se sert de l'inégalité $|x| < \varepsilon$. Le nombre ε est proportionel à l'erreur relative d'arrondissement des réels. Une évaluation de la valeur universelle ε pour le cours de tout le programme est démesurée.

FRACTAL BASED PROCEDURAL MODELLING.

Norbert Filip
Department of Geometry
Faculty of Mathematics and Physics, Comenius University,
842 15 Bratislava, Slovakia
e-mail : filip@delta.dcs.fmph.uniba.sk

1. ABSTRACT.

Two different methods for procedurally based modelling are presented as an alternative to the interactive modelling. Fractal and graftal techniques are shown to be the possibilities for modelling of natural phenomena as terrain shapes and trees. Different issues of consistency for fractal based terrain modelling are discussed. Second part of the paper shows several problems of graftal and L-systems trees modelling. Attention is focused on modelling of branches, the problems of "twisted" branches and branch connections are solved. The paper contains remarks towards the implementation of presented ideas.

2. INTRODUCTION.

There are two ways, how to put data for rendering in to a computer:

- By user. User generates data from basic elementary geometric objects (lines, triangles, spheres, ...) using elementary geometric operation (union, intersection,...).
- Procedurally. Data are generated by procedural algorithms.

This paper describes the methods of procedurally generating data. Very popular and useful part of procedural modelling is based on the fractal techniques. Fractal geometry is very often used in methods of generation of natural (or naturally looking) models. Naturally looking objects cannot be generate by user using elementary operations on basic elements, because they are very coarse and complex. Therefore, fractal geometry, with its properties (self-similarity, subdivision and iterative character...) is very useful to generate models of real objects.

Next chapters deal with two basic fractal based modelling systems.

3. FRACTAL BASED TERRAIN GENERATION.

Example of measurement of England's coastline is very often used as a typical example of fractal object. An algorithm, which produces naturally looking models of coastline (or mountain) in 2D, can be describe as follows.

An algorithm starts with some approximation of a terrain (it is a system of lines) and uses method of subdivision. Subdivision in this context means both splitting the polyline into smaller and smaller parts and, at the same time, spatially perturbing these parts. The initial gross shape of the objects is retained to an extend that depends on the perturbation applied on each subdivision level. The subdivision algorithm most commonly used was developed by Fournier in 1982. This algorithm was introdiced as a cheap alternative to the more rigorous procedures suggested by Mandelbrot. It uses self-similarity and conditional expectation properties of fractional Brownian motion to give an estimate of the increment that is used in the line subdivision.

3.1 TERRAIN GENERATION IN 2D.

Now we describe exactly algorithm of a line subdivision.

ALGORITHM (Fournier in 1982):

1. **create some approximation of a terrain**
2. **while (done)**
   subdividing the lines and spatially perturbing them

The most interesting if problem of subdivision of a line.
Let is perturbed the middle point of the line in the direction of line normal. The recursive formula for subdivision of a line segment can be written as follows [WW92]:

$$x'' = \frac{(x+x')}{2} + \text{roughness} * (y'-y) * R$$

$$y'' = \frac{(y+y')}{2} + \text{roughness} * (x'-x) * R$$

Where :

- **R** is a random number in the range -1 to 1.
- **roughness** determines the extend of the perturbation, it is the parameter of the procedure in which the recursive formulas embedded.
- **(x,y),(x',y')** are coordinates of end points of the line.
- **(x",y")** are coordinates of perturbated midle point of the line.

This formula was developed from relation between sides of similar triangles. (see figure 1)

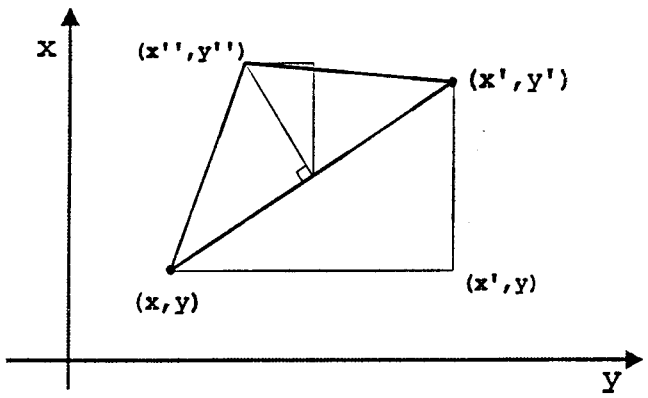
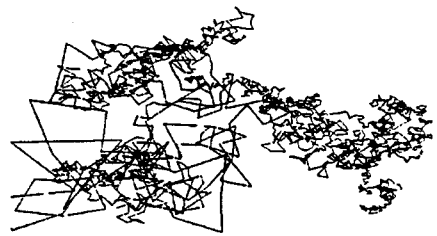


Figure1: Line subdivision in 2D terrain generation.

Note: The perturbation automatically decreases as the recursion level increases, since it is scaled by the length of the line (if is **roughness** defined correctly, it means less then $\sqrt{3}/2$).

roughness=0.3



roughness=1.0 (greater than $\sqrt{3}/2$)

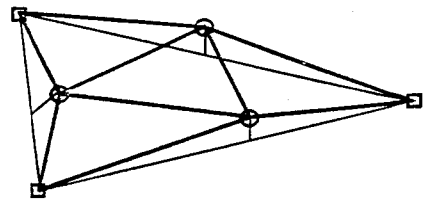
Figure 2: Terrains generated with different roughness.

3.2 GENERATING TERRAIN IN 3D.

Now we describe algorithm to generate terrain in 3D. Let a basic element is a triangle (in 2D it was a line). We start with some coarse approximation to the final landscape (for example a pyramid to simulate a mountain), and then recursively subdivide each basic element (like in 2D). Method of dividing of triangle is better explained using method of dividing a line:

1. **divide all three edges of the triangle and perturbing they in the direction of the normal of the triangle**
2. **create four new triangles and delete the old one**

Next figure makes this algorithm clearer.



○ NEW POINTS
□ ORIGINAL POINTS

Figure 3: Triangle subdivision.

There are two difficulties that emerge from this method. These are categorised by Fournier as **internal and external consistency** problems [WW92].

- **Internal consistency** requires that the shape generated should be the same irrespective of the orientation of the object that is subject to the subdivision. To satisfy internal consistency, the random numbers generated must not be a function of the position of the points, but should be unique to the point itself. This problem can be solved by giving each point a key value that is used to index a Gaussian random number generator.
- **External consistency** is harder to maintain. Within the mesh of triangles every triangle shares each of its sides with another and the same random displacements must be generated for corresponding points of different connecting triangles. This is already solved by using the key value of each point and the hash function, but another problem still exists - that of the direction of the displacement. To prevent gaps opening up, we have to arrange that the direction of displacement out of a shared edge is along a normal that is average of the normals of the polygons that contribute to the edge, or simply method in which is direction of displacement only one (for example up in mountains).

To satisfy external consistency problems it is essential to use more complex data structure, and compute the displacement only ones.

```
struct point3   { double x; double y; double z;};
struct vector3  { double x; double y; double z;};
struct edge     { index_to_point A; struct triangle *triangle1,
                        *triangle2;};
struct triangle { struct edge *a, *b, *c, *is_divideda, *is_dividedb,
                        *is_dividedc; struct triangle *next; struct
                        vector3 normal;};
```

The following is a description of an algorithm of dividing the edge *a* in triangle *T* and *pert_point* is perturbed middle point of edge *a*.

```
If (T.is_divideda != NULL) pert_point = T.is_divideda->A;
 else
 {
  pert_point = comp_pert(T.a->A,T.b->A,(T.a->triangle1->normal +
                T.a->triangle2->normal)/2.);
  set in 2-nd triangle value is_divided on pert_point;
 }
```

where ***comp_pert(A,B,n)*** computes perturbation of middle point of line ***A,B*** in direction ***n***.

3.3 TERMINATION CRITERIA FOR FRACTAL SUBDIVISION.

So far nothing has been said about the termination criteria used for fractal subdivision. Fractal polygons are input to renderer at their coarsest level of detail and the subdivision proceeds within the renderer. Termination occurs when subdivision of the given elements results in a shaded value which is the same as that obtained from shading the element without subdivision. So the renderer when shading a given pixel, has to process all fractal elements that might affect the value of that pixel. Recall that the sum of the perturbation, applied at every level of subdivision, can be thought of as making up a geometric progression that must converge, hence at any given level the fractal can be bounded in space. It is the projection of the bounding box on the screen that marks out what pixels it may affect.

A problem arises in animation if the subdivision criterion is screen driven - for example, subdivide until the area of the polygon is less than one pixel. In this case, when the polygon moves, a situation easily occurs where polygons are subdivided to a different level in different frames. This result in a characteristic bubbling effect in the sequence.

4. FRACTAL BASED TREE GENERATION.

The next very interesting task of computer graphics is generation of models of trees. We will be deal with this topic in the rest of this paper.
Let's try to describe algorithm of tree growth.

Symbol **b** represents a bud, symbol **d** represents a branch, a left parenthesis **[** is the beginning of the branch and right parenthesis **]** represents branch's end. We will proceed year by year . There is only one bud in the first year (**b**). Bud is replaced (rewritten) by the system of branches (there is one new bud on each branch) **b→d[db][db]**, so the new tree can be written in the following form **d[db][db]**. It continues every year (see figure 4).

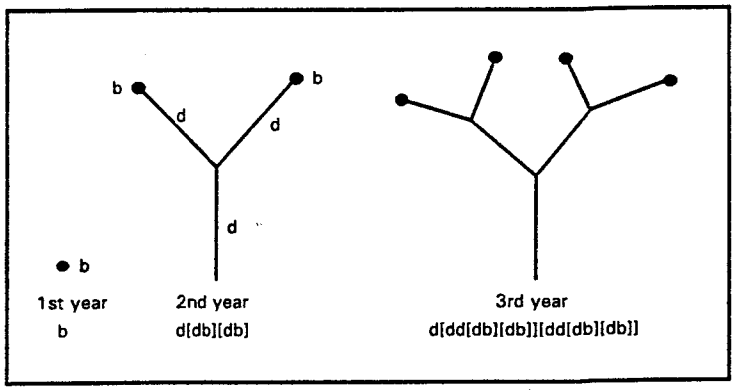


Figure 4: Graphical interpretation of strings.

We can see similarity with a parallel rewriting grammar. It may be true, that the same way of thinking has been used by Aristid Lindenmayer, whose suggestions were a fundamental base for my work. He introduced L-systems, and thus as a consequence, foundation of an axiomatic theory of development have been established.

4.1 L-SYSTEMS.

An example of rewriting system described up is called ***D0L-system*** (D-deterministic, 0-context free, L-Lindenmayer).

L-systems are parallel rewriting grammars. The parallel rewriting grammars are similar to those used in conventional formal language theory, except that the rules are applied simultaneously on a given word. All strings are generating from a initial string called *axiom*. These grammars are called *pL-systems*, where *p* is the number of nearest neighbours used in the context of a symbol in the application of various rules (if *p* is equal to 0, it is a context free grammar).

All plants generated by the same deterministic 0L-systems are identical. Variation can be achieved by randomizing the interpreter, the L-systems, or both. Randomization of the interpretation alone has limited effect. While the geometric aspects of a plant (such as the branch length and width, and the branching angles) are modified, the underlying topology remains unchanged. We can change topology by the ***stochastic 0L-systems***. Stochastic 0L-system's rules have a following form **a —p→b**, where **p** ($0<p\leq 1$) is probability of the rule **a→b**. Stochastic L-systems can be used for modelling of deviation of plants.

Context sensitive L-systems (2L-systems or 1L-systems) are necessary to model information exchange between neighbouring symbols. 2L-system's rules are in the form $l<a>r \rightarrow b$, where the letter **a** can produce string **b** if and only if **a** preceeded by letter **l** and followed by **r**.

There are no information on age of letters (it is nessesary to have this information, because for example the width and length of a branch depends on it's age). This problem can be solved by using the *iterated L-systems*. Rules are in the form $a_i \rightarrow ba_{i+1}c$, where i is age of a letter **a**.

4.2 GRAFTALS.

This method for modelling of trees is called graftal. The graftal consists of two parts: *generator* and *interpreter*. The generator generates string using rewriting rules (L-system), and interpreter interprets the generated string and convert it into an image.

Graphical interpretation is based on the notion of a LOGO-style turtle. Our interpreter is based on 4 points given bellow too:

1. Child branches are generated by one branching (bifurcation).
2. The length and diameter of the child branches become shorter at constant ratios.
3. Child branches lie on the maximal gradient plane of their mother branch. The first child branches in particular lie on the parallel to direction of gravity.
4. Branching occurs simultaneously at the tips of all branches.

But in 3D we can't represent step of the turtle as a line segment, it is necessary to represent it as a three dimensional object.

4.2.1 Three dimensional branch representation

The following notation should be useful: A= a root of the branch, B=a tip of the branch, r=diameter, l=length ,*s*=vector of the branch orientation , and rr=diameter of the mother branch, *rs*=vector of the mother branch orientation.

Let $K(s,A,r,u)$ is a parametrical expression of the circle (central point is A, its diameter is r, circle lies in the plane which is perpendicular to the vector s and u is parameter) and $C(A,rs,B,s,v)$ is a parametrical expression of the parametrical cubical curve (A is a beginning of the curve and rs is a tangent to the curve in point A, B is curve's end point and s is a tangent in B and v is parameter). Thus:

1. $C(A,rs,B,s,0) = A$
2. $C(A,rs,B,s,l) = B$
3. $C'(A,rs,B,s,0) = rs$
4. $C'(A,rs,B,s,l) = s$

Note: It is important, end of the curve has parameter equal to branch length (4th parameter in expression 2 and 4 is l).

Therefore, the surface area of the branch can be parametrically expressed as follows:

$$P(u,v)=K(C'(A,rs,B,s,v),\ C(A,rs,B,s,v),\ r.v+(l-v).rr,\ u)$$

There are some problems concerning with relevant choice of circle parametrisation, the problem of "twisting" boundary curves must be solved.

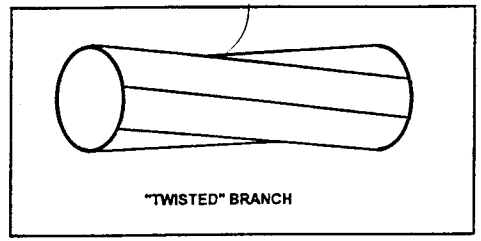


Figure 5: Branch incorrectly parametrised.

Let $K_{u1}(s,A,r,u) = (r.\cos(u+u_1) , r.\sin(u+u_1) , 0) . M_s + A$, where M_s is matrix transforming vector (0, 0, 1) into vector s, be parametrical expression of a circle, which depends on the starting value u_1 (*K*, s, A, r and u have the same sense as above). Let $K_{u1}(rs,A,rr,u)$ and $K_{u2}(s,B,r,u)$ be parametrical expressions of the root and the top of the branch. Value u_1 we know from parametrical expression of previous (mother) branch. Now we must compute value u_2, so that the branch, which is represented by two circles $K_{u1}(rs,A,rr,u)$ and $K_{u2}(s,B,r,u)$, won't twisted. Let v' is projection of vector ($K_{u1}(rs,A,rr,0)$ - A) into the plane of circle $K_{u2}(s,B,r,u)$. If we don't want to have twisted branch is necessary that the angle between vectors v' and $K_{u2}(s,B,r,u)$ is equal to 0. We can compute value u_2 from this equation. If u_0 is angle between vectors v' and $K_0(s,B,r,0)$ then value u_2 is equal to sum u_1 and u_0 ($u_2 = u_1+u_0$).

4.2.2 Botanical rules.

Using graftal methods for modelling realistic images of trees, we shouldn't forget to include some botanical aspects of tree genesis. Some of this are given bellow.

DIVERGENCE ANGLE:

The pattern formed in the stem by the alternation of child branches is affected by the **phyllotaxis.** A common method of expressing the phyllotaxis is by reference to the so-called genetic spiral, which passes through the child branches in their numerical order (the order of their production). The **divergence angle** between the branches is given as a fraction of a circle, which is estimated by finding two superimposed branches and counting the number of branches and the number of turns around the axis between the two superimposed branches. The first value is used as the numerator, the second as the denominator of the fraction. The fractions most frequently belong to the Fibonacci summation series 1/2, 1/3, 2/5, 3/8, etc., in which each value of the numerator and the denominator is a sum of the two corresponding values that precede it.[ES65]

TROPISM:

The next important botanical aspect is a **tropism**. It is a movement of plants or its parts provoked by one-orientation impulse (e. g. phototropism, geotropism etc.). Tropism can be mathematically expressed by of tropism impulse vector (v) and its power (f). If s is branch's orientation then we can expressed s' (new orientation of the branch, it means orientation after considering of a tropism) as follows: $s'=1/(1+f).(s+f.v)$, where $0<f\leq 1$.

5 PROGRAM REALISATION.

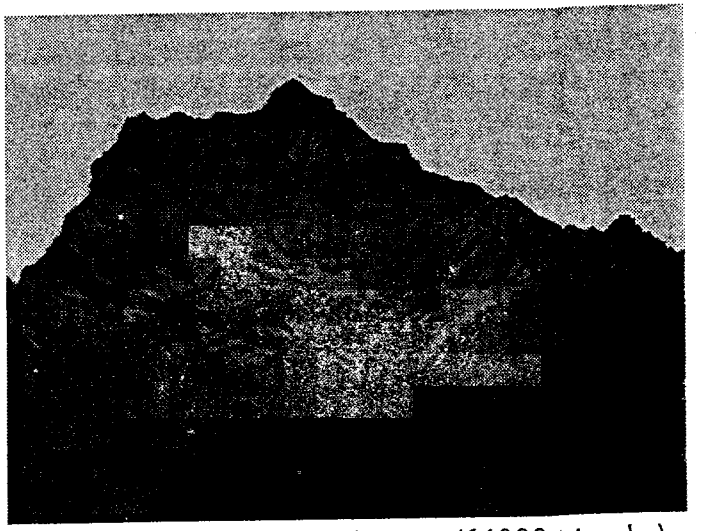
We have implemented a couple of algorithms for terrain and tree generation in C language on PC AT. Model can be imported into CAD programs (by DXF files), or into 3DSTUDIO (by ASC files). Models can be rendered (by Gouraud's local illumination model and visibility is computed by z-buffer algorithm. The tree model creating time is about 80 sec. (tree with about 1500 branches -3.5MB file) and rendering time is about 90 sec. on AT/486/50Mh. Model of terrain was created on HP Apollo 700 as well.

6 SOURCES.

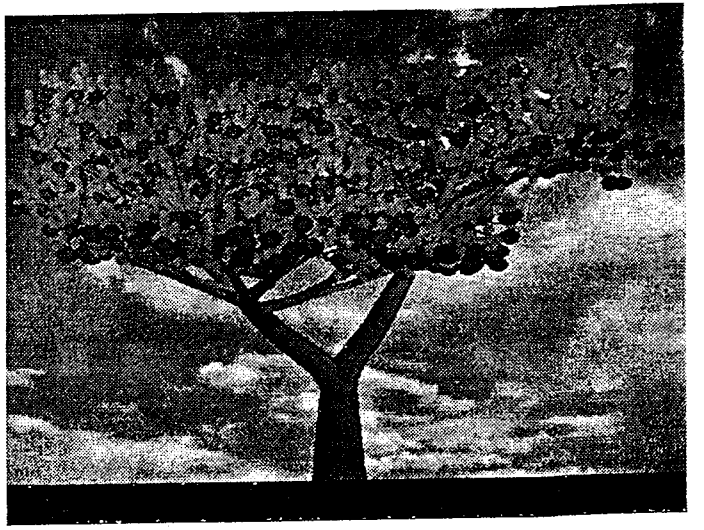
- [WW92] WATT, MARK WATT: **ADVANCED ANIMATION AND RENDERING TECHNIQUES**, ACM Press 1992
- [PR89] PRZEMYSLAW PRUSINKIEWICZ, JAMES HANAH: **LINDENMAYER SYSTEMS FRACTALS AND PLANTS**, vol.. 79 of Lecture notes in biomathematics, Springer verlag, Berlin 1989
- [AO84] MASAKI AONO, TOSIYASU L. KUNII : **BOTANICAL TREE IMAGE GENERATION**, IEEE Computer Graphics & Applications, may 1984, pp. 10-33.
- [GA90] ANDRE GAGALOVICS: **SPECIAL MODELINGS** , Tutorial Note, EUROGRAPHICS 1990
- [FI93] NORBERT FILIP: **MODELOVANIE STROMOV POMOCOU L-SYSTÉMOV**, Diploma Thesis, MFF UK Bratislava 1993
- [ES65] KATHERINE ESAU : **PLANT ANATOMY**, John Wiley & Sons, Inc. , 1965
- [CH66] MICHAEL CHINERY : **A SCIENCE DICTIONARY OF THE PLANT WORD**, Sampson Low 1966

APPENDIX

Results of the presented methods



3D procedurally generated terrain (64000 triangles)



Graftal tree with about 3000 branches