RAY TRACING OF NONLINEAR FRACTALS

Peter Wonka **Micheal Gervautz**

Technical University Vienna
Institute for Computer Graphics
Karlsplatz 13/186/2, A-1040 Vienna, Austria
email: {wonka|gervautz}@cg.tuwien.ac.at

ABSTRACT

We present a new kind of parametric Lindenmayer-systems called nonlinear CSG-pL-Systems, which are useful for the modeling of nonlinear fractals and fractal like objects. Nonlinear CSG-pL-systems describe cyclic CSG-Graphs, which can include several nonlinear transformations. To render the given objects a ray tracing algorithm is introduced, that is independent of the transformations and the under laying CSG-primitives. We use tapering, twist and bend as nonlinear transformations for our implementation. The new modeling possibilities and their visualization for abstract fractals and natural phenomena are investigated.

KEYWORDS: nonlinear fractals, nonlinear ray tracing, CSG-pL-systems, natural phenomena.

1 INTRODUCTION

Fractals are a powerful and memory saving approach to model complex three dimensional objects. With the short description of a fractal it is possible to generate a geometric structure of an arbitrary amount of detail, that could hardly be described with an enumerative approach. To make fractals a useful modeling tool, it is necessary to use a flexible, parameter controlled class of fractals, which makes it possible to have sufficient control over the outcome. Since nonlinear transformations have been proven to be intuitive and convenient aids in the modeling process, it is our goal to develop a modeling tool that combines the advantages of nonlinear transformations and fractal modeling.

One observation that can be made when studying natural phenomena like plants is the highly complex structure and the strong degree of nonlinearity of the geometry. For the modeling of natural phenomena procedural modeling techniques are the most adequate and produced the best results up until now. Therefore we want to use the nonlinear CSG-pL-Systems for the modeling of plants. In order to get a high quality picture the objects will be rendered with ray tracing.

In the past Demko, Hodges and Naylor [DHN85] used iterated function systems(IFS) for the definition of complex objects. Although they limited their approach to two dimensional objects their results looked very promising. Ray tracing of three dimensional IFS was investigated by Hart and DeFanti [HD91] using the concept of object instancing to construct the fractals dynamically during rendering. Nonlinear transformed IFS were investigated by Zair [ZT96] using free form techniques and Gröller [Gro94] presented a method for modeling and rendering nonlinear IFS. The problem with IFS modeling is that the constructed objects are strictly self similar, which is too restrictive to make them useful for the modeling of real world objects. Another concept which is adequate for the description of these objects are L-Systems which were introduced 1968 by A. Lindenmayer [Lin68]. Their use for the modeling of plants was shown by Prusinkiewicz and Lindenmayer [PLH$^+$90] with several extensions like parameter passing, stochastic behavior and context sensitive rules. The limitation of their approach is the rendering technique which uses huge amounts of memory, because the scene description is built up in advance, which also makes nonlinear extensions difficult. A solution to overcome the memory limitation are CSG-pL-Systems [GT96], which provide similar modeling possibilities and provide a useful basis for nonlinear extensions. Therefore we use CSG-pL-systems as a basis for our approach to include nonlinear transformations in fractal modeling.

2 BACKGROUND

CSG-pL-systems [GT96] are parametric string rewriting systems that can be represented by cyclic CSG-graphs, which need much less memory than the explicit CSG-expressions. The parameters are treated like global variables, so that any node can access all the parameters it needs. A graph consist of the following elements:

- The leaves of the CSG-graphs are geometric **primitives** like cylinder, box or sphere.
- Sub-graphs are combined with the **CSG-operators** $op \in \{\backslash, \cup, \cap\}$.
- Affine transformations are described in transformation-nodes (**T-nodes**), so that they can be applied to primitives and whole sub-graphs as well. A T-Node consists of a transformation list, so that several transformations can be stored in one node. The transformations can depend on certain parameters. e.g. scale $(a,0,2b)$.
- Calculation nodes (**C-nodes**) are used to modify the parameters. New values can be assigned to any parameter according to mathematical expressions. This is very similar to the assignment of a value in a programming language. e.g. a=a*a+2*c.
- Selection-nodes (**S-nodes**) are used to control the construction of the CSG-expression and have more than one successor in the graph. They select the actual successor based on a boolean expression. The boolean expression depends on given parameters. S-nodes behave like if-then-else statements: e.g. if $(b \leq 3 * c)$ then select node1 else node2.

CSG-pL-systems describing smaller objects can be rendered interactively with Z-buffering to obtain a quick result [SG97] during modeling. For high quality pictures they are used directly for ray tracing.

Figure 1 shows an simple example graph representing a Sierpinski-tetrahedron. Beside IFS CSG-pL-systems are mainly used to describe complex models like plants, shells and architectural objects.

3 NONLINEAR EXTENSION

For our system design we have to consider two major points: modeling and rendering. The use of nonlinear transformations should enrich the range of possible models without making the design of new objects too complicated. For the rendering part we choose ray tracing, because it produces highly realistic images especially for outdoor scenes. These are our design goals:

- It should be possible to apply nonlinear transformations to geometric primitives and whole CSG-graphs as well.
- The used class of transformations should be an intuitive modeling aid.
- The transformations should fit into the concept of ray tracing CSG-graphs.
- The computational effort should stay in an acceptable range.

Tapering, Twist and Bend [Bar84] were proven to be very useful to describe the structure of natural phenomena like plants. Branches for example are almost always bended and tapering can be used to modify the thickness of the branches. Oppenheimer [Opp86] successfully used the concept of tapering and twist for the creation of complex tree models. The result of tapering and twist applied to a twig can be seen in figure 2. Therefore we extended CSG-pL-systems to allow these three transformations. Nonlinear transformations can be used in transformation nodes, together with linear transformations. In this way whole CSG-graphs can be transformed with one nonlinear transformation.

4 RENDERING ALGORITHM

The idea of ray tracing three dimensional fractals is to use the concept of object instancing to construct only parts of the fractals dynamically during rendering. The major advantage of this approach is to render fractals with very low memory consumptions. Hart [HD91] presented this approach for IFS and Gröller [Gro94] extended this concept for ray tracing nonlinear IFS.

The ray tracing algorithm for linear CSG-pL-systems uses a ray transformation instead of an

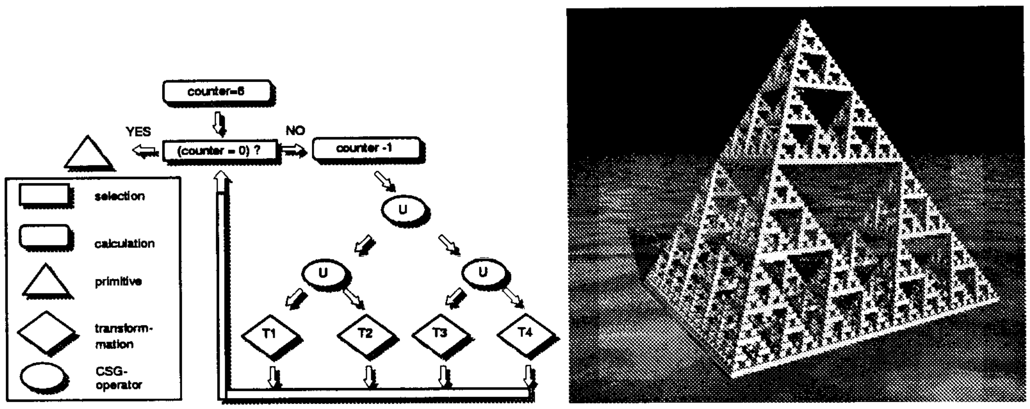

Figure 1: The CSG-graph on the left represents the Sierpinski-tetrahedron and can be used for ray tracing(right picture).

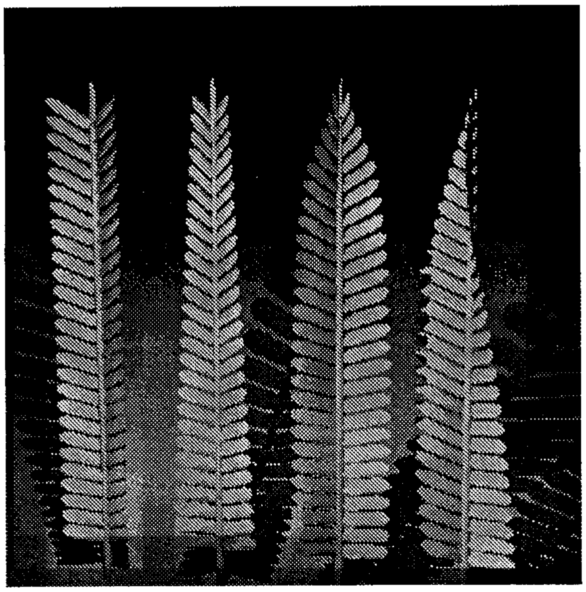

Figure 2: A set of twigs created with tapering and twist applied to a simple twig model.

object transformation to calculate the ray-object intersections. To calculate an intersection point with an object represented by a CSG-Graph the ray traverses the graph. When a transformation is encountered the ray has to be transformed so that ray-primitive intersections can be calculated in the local coordinate system of the primitives.

To keep the advantage of object instancing it is necessary to transform a ray immediately whenever it encounters a nonlinear transformation. This transforms the linear ray into a three dimensional curve, which makes the representation of the ray consisting of an eye and direction vector no longer sufficient. Another representation has to be found.

The main idea is to use a piece by piece linear approximation as proposed by Gröller [Gro95].

Within a transformation area the ray is split into N linear segments defined by $N+1$ points p_i. Since it is hard to determine how many segments are necessary for an accurate representation, an error metric has to be found which guarantees, that the maximum distance of the linear pieces to the analytically correct ray does not cause noticeable errors in the picture. To maintain a certain accuracy without time intensive computations adaptive refinement is used.

4.1 Adaptive Refinement

Given a nonlinear transformation f_{nl}, the ray defined by $p_i, \quad 1 \leq i \leq N+1$ is transformed with the inverse transformation f_{nl}^{-1}. After this transformation the accuracy of the representation does not hold the error bound in general and differs for different ray segments. To improve the approximation for every three points p_i, p_{i+1} and p_{i+2} the maximum error is estimated with the distance from the point $f_{nl}^{-1}(p_{i+1})$ to the chord $\overline{f_{nl}^{-1}(p_i) f_{nl}^{-1}(p_{i+2})}$. This is the height of the triangle defined by the three points. If this distance exceeds a defined threshold new points are inserted between p_i, p_{i+1} and between p_{i+1}, p_{i+2}. The refinement takes place until the condition is met for all points. The process of the adaptive refinement is illustrated in figure 3. As the ray traverses the graph down to a primitive it passes many nonlinear transformations. Therefore the process of adaptive refinement has to be repeated after each transformation.

In our implementation a T-node consists of a list of m linear or nonlinear transformations f_i. The transformation of a whole T-node is defined by $f_{T-node} = (f_1 \circ f_2 \cdots \circ f_m)$. Adaptive refinement is done once for the transformation f_{T-node} instead of calculating each transformation f_i separately, gaining speed.

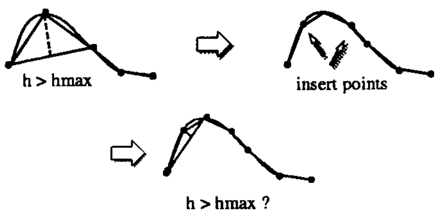

Figure 3: Adaptive-refinement of two ray segments. If the height of the triangle defined by the three points is greater than a certain threshold, the ray segments are refined.

For our final pictures we used a threshold of 10^{-5}units for objects which are about 1 units big. This conservative estimation was empirically found because of test measurements. A maximum bound for the occurring error cannot be given with this method.

4.2 Ray-primitive intersection

As the nonlinear ray consists of linear segments the ray-primitive intersection can be calculated by successively intersecting a single segment with the primitive using existing linear intersection algorithms. This means all primitives of a ray tracer can be used in combination with nonlinear transformations. In many scenes only union-CSG operators are used. For these cases an additional speedup can be achieved. The calculation can stop after the first intersecting segment, because no further intersections can be visible.

4.3 Normal vector calculation

To handle nonlinear transformations in ray tracing it is necessary to know not only the transformation function but also the inverse transformation function and the normal vector transformation. Tapering and twist, in the most general case, use an arbitrary function $f : R \to R$ to control the transformation. For the calculation of the normal vector transformation the first derivation of this function is needed. Numerical calculations during the rendering process are much too expensive. Therefore only certain transformations are used, where the first derivation can be calculated easily in advance. For the most cases a piece by piece linear function is sufficient.

5 IMPROVEMENTS

5.1 Bounding Boxes

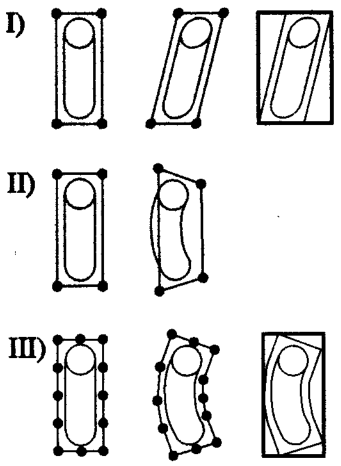

Figure 4: This figure shows the bounding box problem for 2 dimensions: I) A cylinder is transformed with linear transformations. The 4 points are sufficient for a bounding box calculation after the transformation. II) The cylinder is bended and the cylinder is no longer surrounded by the bounding box determined by the four points. III) The solution is to use additional points for the bounding box calculation.

Bounding boxes are very essential for the performance of the ray tracing algorithm. They provide two advantages:

- Bounding Boxes reduce significantly the intersection calculation for hierarchical structures.
- Bounding boxes guarantee a finite transformation area. The nonlinear ray can be clipped with each bounding box reducing the number of ray segments to be considered for further calculations.

The bounding box calculation for linear CSG-pL-systems was investigated by Traxler et al. [TG95]. Their approach used hyper bounding boxes for each node of the CSG-Graph instead for each instance of each node. Traxler used eight points to approximate the convex hull of primitives. These eight points together with all transformations along the path to the primitive can be used to calculate the contribution of a primitive to the bounding box of an intermediate node. Their method calculates bounding boxes which are tight enough for ray tracing.

To adapt this algorithm for nonlinear CSG-graphs we have to consider, that a set of 8 points might no longer be sufficient to approximate the convex hull for a nonlinear transformed primitive (see figure 4). To solve this problem we extended the point set to form a mesh defined by the original box.

These points are used as the new approximation of the convex hull. With the new approximation it is very unlikely, that a part of the transformed primitive lies far outside the transformed point set. However even a fine mesh cannot guarantee a correct bounding; small parts of the object may lie outside the bended mesh.

Therefore the point set of the original bounding box is scaled up a bit before the fine mesh is constructed. In this way we reduce the probability that parts of the object lie outside the bounding mesh after transformation.

This approach satisfies two important conditions: The calculation works exact enough, so that no bounding errors occurred in our tests and the computation is inexpensive compared to the rendering time.

5.2 Strip Trees

The straightforward approach to successively intersect each segment of the ray with a bounding box works well, but can further be improved. A faster solution would be a hierarchical handling of the ray. We use strip trees to subdivide nonlinear rays and create hierarchical bounding boxes for them.

The strip tree was introduced by Ballard [Bal81] and first used by Kajiya for ray tracing [Kaj83]. The strip tree used in our approach is quite similar to the strip tree proposed by Gröller [Gro95], who uses axis aligned bounding boxes for a three dimensional ray described by an explicit mathematical function.

We will also use axis aligned boxes for our ray. Since our ray is represented by linear segments the construction of the strip tree is straightforward and very fast.

The leaves of the tree are linear ray segments bound by a box. The inner nodes of the tree combine the boxes of their successors. To construct the tree the bounding boxes are combined bottom-up from the leaves up to the top.

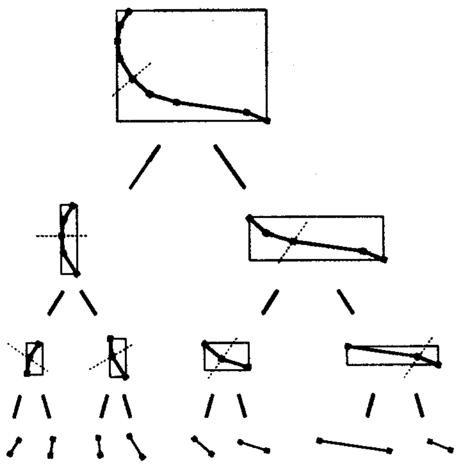
Figure 5: The figure shows a strip tree for 8 ray segments.

For the ray-object bounding box intersection test it is important to note, that we do not need the exact intersection point. It is enough to determine which segments of the nonlinear ray are completely outside the object bounding box and clip them. For this calculation the strip tree is tested against the object bounding box. Starting with the root node, the bounding box of the strip tree node is intersected with the object bounding box. Three configurations are possible:

- The bounding box of the strip tree node is inside the object bounding box. In this case all segments of the ray are considered for further calculations.
- The boxes have no overlapping area. This means that the parts of the ray bound by the strip tree node can be clipped.
- The boxes partly overlap. In this case the object bounding box is tested against the two successors of the strip tree node. If the considered strip tree node is a leaf node the ray segment can not be clipped .

Traversing down the graph this way, only ray segments reach a primitive which lie inside the primitive bounding box.

6 MODELING

We used nonlinear CSG-pL-systems for the modeling of abstract fractals and plants. Since nonlinear IFS are a subset of nonlinear CSG-pL-systems we can render nonlinear IFS and nonlinear transformed IFS. The use of nonlinear transformations gave us several advantages:

- The modeling of complex transformations applied to a whole graph is possible. We can take any object represented through a CSG-Graph and apply a nonlinear transformation to it.
- For complicated nonlinear geometry it is sometimes too expensive to approximate it with linear transformed primitives. A strongly twisted stem for example would need a large amount of primitives for an adequate representation.
- A great variety of similar objects can be produced by applying different nonlinear transformations to them. An example can be seen in picture 2.

7 RESULTS

Nonlinear CSG-pL-systems were implemented as an extension to the well known free-ware ray tracer Povray. The implementation is very modular and uses the core functions of POV 3.0 so that nearly all primitives and features offered by Povray can be used. Nonlinear fractals can be seen in the pictures 6 and 7. They show the use of nonlinear transformations in each level of recursion. Both fractals were produced by a variation of the Sierpinski tetrahedron definition described in figure 1. Nonlinear tapering is applied within the recursive description of the fractal. In figure 6 an additional tapering is used to transform the whole object. Picture 8 shows the construction of palms and the hut in figure 9 demonstrates the use of twist for the construction of climbing plants. The plants are constructed with linear transformations and a twist is applied to obtain the final geometry. The memory requirements for all scenes are under 100 Kbyte.

8 CONCLUSION

Nonlinear transformations were used to create a new kind of pL-Systems called nonlinear CSG-pL-systems. We showed that the given system is a useful modeling tool for abstract fractals and natural phenomena. Our experience was, that the ray tracing algorithm and the bounding box calculations are very stable and even extreme transformations do not result in visible errors in the final picture. But one has to be cautious using nonlinear transformations for the modeling of real world objects because transformations applied to a whole CSG-Graph might result in unwanted side effects. The leaves in picture 9 for example are all twisted around the stem. This is not a problem for the shown picture, but for close views the modeling artifacts become visible.

9 ACKNOWLEDGMENTS

Thanks to Christoph Traxler for his useful comments on that project and Wolfgang Deutsch for parts of the implementation.

References

[Bal81] Dana H. Ballard. Strip trees: A hierarchal representation for curves. *Communications of the ACM*, 24(5):310–321, May 1981.

[Bar84] A. H. Barr. Global and local deformations of solid primitives. In H. Christiansen, editor, *SIGGRAPH '84 Conference Proceedings (Minneapolis, MN, July 23-27, 1984)*, pages 21–31. ACM, July 1984.

[DHN85] S. Demko, L. Hodges, and B. Naylor. Construction of fractal objects with iterated function systems. *Computer Graphics*, 19(3):271–278, July 1985.

[Gro94] E. Groeller. Modeling and rendering of nonlinear iterated function systems. *Computer & Graphics*, 18(5):739–748, 1994.

[Gro95] E. Groeller. Nonlinear ray tracing: visualizing strange worlds. *The Visual Computer*, 95(11):263–274, 1995.

[GT96] M. Gervautz and C. Traxler. Representation and realistic rendering of natural phenomena with cyclic CSG graphs. *The Visual Computer*, 12(2):62–71, 1996. ISSN 0178-2789.

[HD91] John C. Hart and Thomas A. DeFanti. Efficient anti-aliased rendering of 3D linear fractals. In Thomas W. Sederberg, editor, *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics (SIGGRAPH '91)*, pages 91–100, Las Vegas, Nevada, USA, July 1991. ACM Press.

[Kaj83] James T. Kajiya. New techniques for ray tracing procedurally defined objects. *Computer Graphics*, 17(3):91–102, July 1983.

[Lin68] Aristid Lindenmayer. Mathematical models for cellular interaction in development parts i and ii. *Journal of Theoretical Biology*, 18:280–315, 1968.

[Opp86] P. E. Oppenheimer. Real time design and animation of fractal plants and trees. *Computer Graphics*, 20(4):55–64, August 1986.

[PLH$^+$90] Przemyslaw Prusinkiewicz, Aristid Lindenmayer, James S. Hanan, et al. *The Algorithmic Beauty of Plants.* Springer-Verlag, 1990.

[SG97] D. Schmalstieg and M. Gervautz. Modeling and rendering of outdoor scenes for distributed virtual enviroments. In *Proceedings of ACM Symposium on Virtual Reality Software and Technology 1997 (VRST'97) ,Lausanne, Switzerland,Sep. 15-17, 1997)*, pages 209–216. ACM, September 1997.

[TG95] C. Traxler and M. Gervautz. Calculation of tight bounding volumes for cyclic csg-graphs. *Proceedings of 11th Spring Conference on Computer Graphics, Bratislava*, 11, 1995.

[ZT96] C. E. Zair and E. Tosan. Fractal modeling using free form techniques. *Computer Graphics Forum*, 15(3):C269–C278, September 1996.

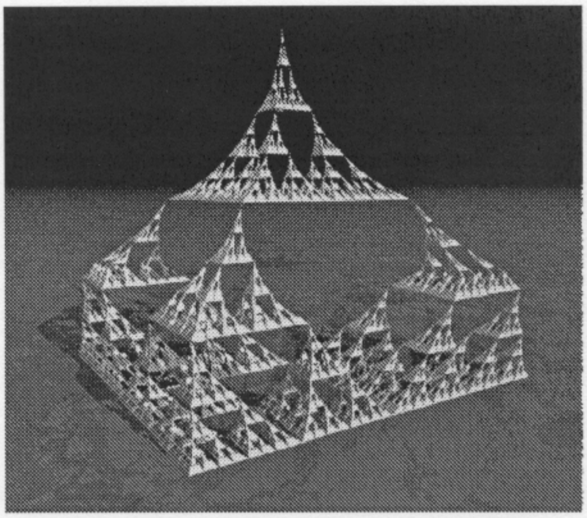
Figure 6: Nonlinear IFS that was transformed with one additional tapering. The rendering of this object took about two hours (A comparable linear version would need 40-60 minutes).

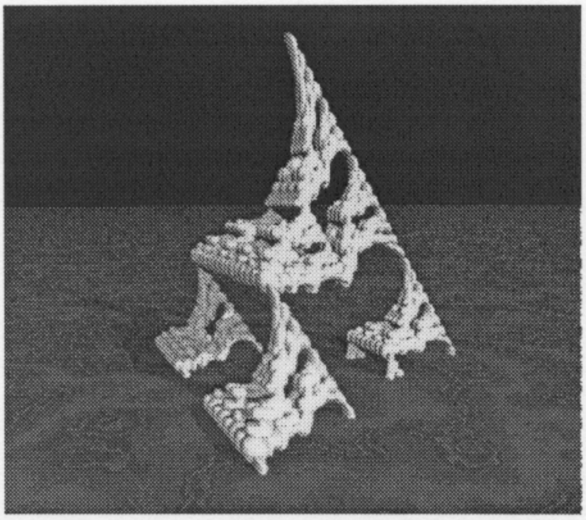
Figure 7: Nonlinear IFS that was transformed with one additional tapering (120-180 minutes rendering time).

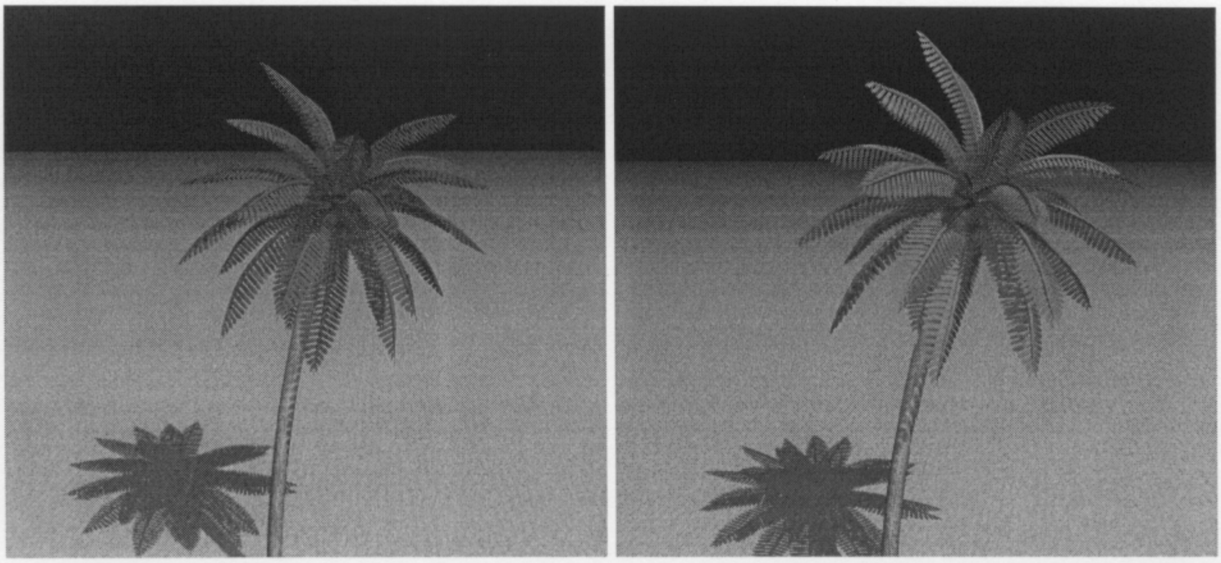

Figure 8: These palms were created using different twigs from picture 2.

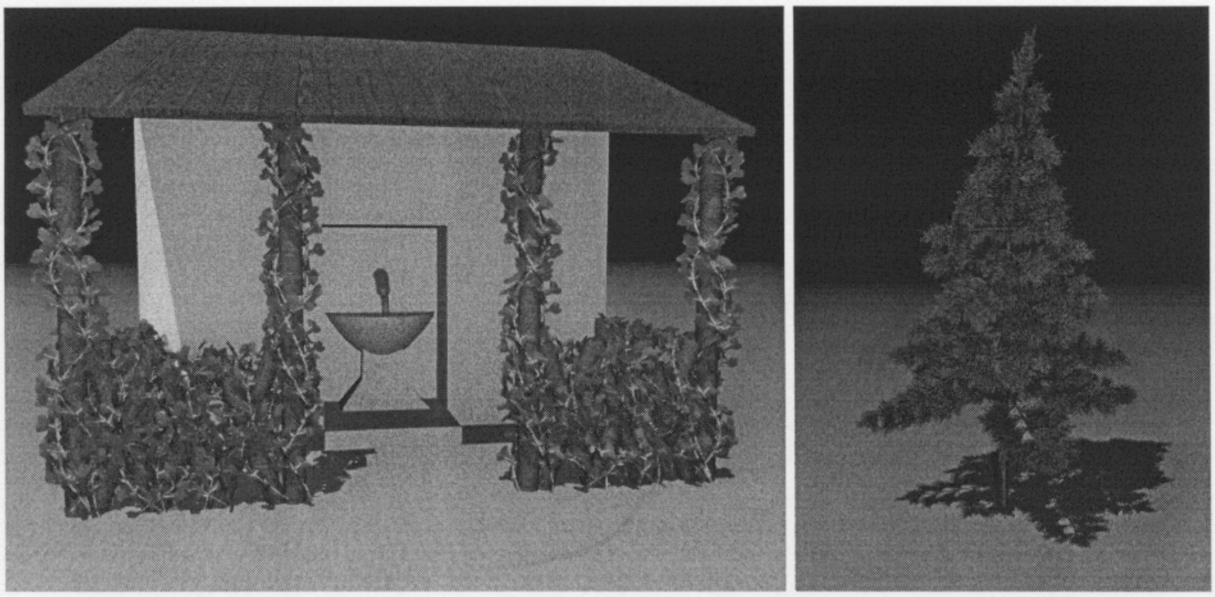

Figure 9: The left picture shows a fence full of climbing plants. The plants and the fence together are represented as one nonlinear CSG-pL-system (about 300 minutes rendering time). The right picture shows a conifer.