Creation and Rendering of Realistic Trees

Jason Weber[1]
Teletronics International, Inc.

Joseph Penn[2]
Army Research Laboratory

"*From such small beginnings - a mere grain of dust, as it were - do mighty trees take their rise.*"
Henry David Thoreau from "Faith in a Seed"

ABSTRACT

Recent advances in computer graphics have produced images approaching the elusive goal of photorealism. Since many natural objects are so complex and detailed, they are often not rendered with convincing fidelity due to the difficulties in succinctly defining and efficiently rendering their geometry. With the increased demand of future simulation and virtual reality applications, the production of realistic natural-looking background objects will become increasingly more important.

We present a model to create and render trees. Our emphasis is on the overall geometrical structure of the tree and not a strict adherence to botanical principles. Since the model must be utilized by general users, it does not require any knowledge beyond the principles of basic geometry. We also explain a method to seamlessly degrade the tree geometry at long ranges to optimize the drawing of large quantities of trees in forested areas.

1 INTRODUCTION

Historically, much of the effort in computer graphics has been directed toward rendering precisely defined geometrical shapes such as manufactured objects whose geometry must be clear-cut and well-defined. CAD tools that are often used to design these objects can also be used to specify the geometrical properties in terms of simpler surfaces or solid geometric primitives. The complexity of many objects is generally low enough to allow complex lighting and ray-tracing computations that approach photorealism.

Natural objects offer a more profound challenge. A hillside may contain hundreds of trees, billions of grass blades, and countless rocks, pebbles, and ground variations. Each tree may easily be characterized by hundreds of thousands of leaves and thousands of branches, branchlets, and stems oriented in complex directions. A complex landscape could require an unimaginable large number of polygons to define every minute facet. As a result, complex natural backgrounds containing vegetation are often neglected in high quality image generation and scene simulation because of the difficulty of properly defining and rendering them in a reasonable time. Emphasis is placed on the buildings, vehicles, and assorted manufactured objects that are often the focus of the dominant action in a scene. Because of speed requirements, two-dimensional texture-mapped trees drawn as rotating billboards are common today in many real-time applications, but their appearance can be objectionable. This is especially evident when a viewer is in motion. See [ROHL94] for examples of 2D trees. As simulations become more realistic, the deficiencies in the background objects become more apparent.

We present a model to create and render trees. In designing this model, we have set guidelines focused on the requirements of scene simulation. The foremost requirement is the appropriate level of resolution and quality. For items to appear realistic in a dynamic simulation, the viewer must get the proper sense of rotational as well as translational motion when passing or circling objects.

Realism also depends on the accuracy of textural effects due to leaves and branches within the tree shadowing each other at various times of the day. Therefore, all trees must be three-dimensional. Fortunately, as background objects, trees would rarely be taller than 5 to 20 percent of screen height. Therefore, fine details such as leaf curvature and vein structure are not important. But, a tree's branch structure must be very accurate at this resolution. Leaves do not completely conceal the underlying branches of dormant or sparsely foliated trees.

1 weber@teleport.com, now employed at Dynamics Research Corporation
2 joseph@belvior-arl-irisgt.army.mil

The model must be capable of creating a wide variety of actual tree types and related vegetation such as shrubs, bushes, and palms, as well as cacti and even large grasses. Shrubs, for example, can be easily simulated with the model since they really only differ from trees in that they are usually shorter and have multiple trunks originating directly from the ground [REIL91]. The model must be able to handle random parameters so that a very large number of structural variations can be generated from the design specifications of a particular tree species. It should also implement time-dependent oscillations due to wind and other perturbations.

Use of the model must be understandable by a common user with only a general knowledge of basic geometry, such as directly observable angles and lengths. This excludes the use of any model parameters requiring understanding of difficult principles such as differential equations. Likewise, the model must be stable and easy to use. User-entered free-form equations could easily cause unpredictable behavior. Aspects of the model that may be difficult to control should be isolated from the user and be represented by intuitive parameters. However, the model should not be constrained in a way that interferes with the user's freedom of design.

The specification for the tree must be compact and be able to recreate and render the tree geometry efficiently. This includes the ability to degrade geometry to low resolution at long ranges, where increased speed is necessary to render large forested areas. Any degradation must use negligible overhead and be seamless, even in dynamic simulations where ranges to trees are continuously changing.

This model was designed to successfully meet these criteria. We demonstrate the tree model in our natural environment scene generator. We have developed a compact but varied library of specification files for generating trees that are used in simulating a wide variety of landscapes.

The following section briefly discusses other tree models. The third section gives an overview of our observations of trees. The fourth section goes into the specific details and equations explaining how our parameters are used to create the geometric description. The fifth section explains our method of drawing optimally-degraded instances of the trees at longer ranges. The sixth section is a very short description of our project and how we use the trees in our application. An appendix includes a listing of our parameters and four sample tree specifications.

2 PREVIOUS MODELS

We will make some comparisons and contrasts to other tree models here and throughout the paper. We cannot fully explain the previous work in this space and will direct the reader to definitive references.

Honda introduced a model using parameters to define the skeleton of a tree [HOND71]. He clearly illustrated the difference between the monopodial and dichotomous branching. In dichotomous branching, the branches tend to split apart in different directions from the original. Monopodial branching tends to act similarly except that one branch continues inline from the original. Honda assumes that monopodial branching is a special case concerning structures that are parallel to the line of gravity.

Lindenmayer introduced a string rewriting system [LIND68] for cellular interaction commonly called the L-system. This is later applied to plants and trees and is extensively described in his book with Prusinkiewicz [PRUS90] which describes the system with a few extensions, such as allowing for context-sensitivity and random variations. Basically, the string starts with a seed of a single character. A set of rules defines how to substitute characters during an iteration of rewriting. Presumably, any one character may be converted into several characters. This process is continued iteratively and the string grows. After a designated number of iterations, these strings can be interpreted as geometric commands. Rules can be selected to produce the monopodial or dichotomous branching, as desired.

Aono and Kunii stated that the L-system was not capable of producing complex three-dimensional patterns of branching [AONO84]. They demonstrated their models which also introduced interesting features such as attraction, inhibition, and statistical variations of angles. They made a

detailed evaluation of the arrangement of branches or leaves on a parent stem. Prusinkiewicz and Lindenmayer's book, printed later, argued that Aono and Kunii's rejection of the L-system was no longer justified based on their recent improvements.

Oppenheimer used fractals to form trees. He used parameters such as branching angle, branch-to-parent size ratio, stem taper rates, helical twist, and branches per stem segment. These specifications resemble our approach. The Oppenheimer model, however, following the fractal theory of self-similarity, uses the same specifications for every recursive level. He introduces random variations to alleviate some of the self-similarity [OPPE86]. We believe the self-similarity of fractals to be an unnecessary constraint that limits models to a relatively small number of basic trees. Oppenheimer's images appear to be influenced by Bloomenthal whose paper concentrated on the quality of the surface geometry assuming that a reasonable tree skeleton exists [BLOO85]. Bloomenthal splined between points on the skeleton and used a ramiform to represent branch splitting. He also used a bark texture map created from a digitized x-ray of a plaster cast. However, such detail is only useful when the tree is viewed at very close ranges.

Reeves and Blau created trees and grasses by utilizing a particle system [REEV85]. They primarily emphasized the forest environment instead of concentrating on the structural detail of individual plants. In addition, they decided to focus more on the visual results than the specific details of actual botanical data. For our application, we followed similar guidelines.

De Reffye et al. have had impressive results with a strict botanical model [REFF88]. Their system models growth to a certain age using probabilities of death, pause, ramification, and reiteration. They admit that it takes a considerable knowledge of both botany and of their model to create images with great fidelity to nature.

Since all the models strive to achieve the same result, realistic trees, they will all have some characteristics in common. Although our model does not draw from any of the previous models, comparisons will be made for the benefit of the reader.

3 APPROACH / OVERVIEW

We visualize the structure of a tree as a primary trunk consisting of a variably curved structure similar to a cone. In some trees, this single structure may split multiple times along its length, forming addtional similarly curved structures which can likewise split along their length [CHND88]. This is how we visualize dichotomous branching. The attributes of these "clones" closely match that of the remaining length of their twin, except that they are generated using different random seeds. After splitting, some will tend to curve more to compensate for the directional change caused by the splitting angle.

Monopodial or "child" branches are formed from the trunk and any existing clones. These branches can have entirely different attributes from their "parents". Many attributes, such as length, are defined relative to the corresponding attribute of their parents. For example, a child branch's length is specified as a fraction of its parent's length. These branches, themselves, can have sub-branches and so on. For the resolution requirements of simulation, these levels of recursion can be generally be limited to three or four. It is important to point out that nearly all of the other models consider each branching, whether monopodial or dichotomous, to be discrete levels. They often require nine or ten of these levels. While this is primarily convention, it will be significant in optimized rendering (Section 5). Also, branch level control can assist in designing a tree. We usually begin by deactivating the rendering of all levels but the first (the trunk). Once the trunk's appearance is acceptable, we activate and design the second level, and so on, ascending degrees of complexity to the third and fourth levels. This allows us to view the general shape and structure of the tree without the visual confusion and performance loss due to drawing minor branches and leaves. In many cases, foliated trees can be drawn reasonably well in a final rendering without displaying any of the minor branches.

Specific trees appear to form particular shapes [CHND88,CHAN82]. These shapes are usually the result of the lengths of the primary branches according to their position on the trunk of the tree, e.g., a conically shaped tree has larger main branches near the base of the trunk. Alternatively, it is sometimes easier to define the general shape of the crown by envisioning an invisible envelope around the tree which inhibits growth of branches. In addition, many trees have branches that show a preference to curve towards a vertical direction, either up or down, presumably responding to the competing influences of light and gravity.

Cross-sectional variations can be particularly noticeable in the trunk. The scale of the cross-section does not necessarily taper linearly as with a perfect cone. Some cacti can even have periodic scaling in addition to simple random variations. The radial distance about any particular cross-section can also vary randomly and/or periodically. In addition, the radius of the trunk clearly flares at the base of the many trees.

Wind causes complex oscillatory motion throughout the tree that varies in amplitude and frequency determined by the length and thickness of the trunk and branches.

These are the characteristics we have observed and incorporated into our model. We model enough of the significant effects that a great variety of trees and related objects can be incorporated into any simulation that requires natural environments. Plate 1 shows twenty-four trees rendered with the model.

4 TREE CREATION

The appendix lists most of the parameters currently used by our tree model. It will be used for reference throughout this paper. For the benefit of readers who wish to experiment with the demo program, the intuitive multi-character variable names used in the parameter files will also be used in the equations throughout this paper. We should stress that many of these parameters have standard botanical names which we have neglected. We are not trying to create a new convention, but merely attempting to clarify the meanings of the parameters using simple geometric names recognized by our potential end users. Many of the parameters are repeated for each level of recursion to permit greater control and flexibility. Additional parameters, mostly dealing with seasonal color and lighting properties, are not listed and will not be discussed. The parameters are referred to in the text by name and appear as bold italic, as in ***Shape***. Where necessary, parameters are prefixed by a number that distinguishes similar parameters at different levels of recursion. Generalized parameters can appear in the text with an ***n*** prefix, such as ***nTaper*** referring to ***0Taper***, ***1Taper***, ***2Taper*** and ***3Taper***. This refers non-specifically to any of the like parameters. Many parameters are followed by a variation parameter with the same name and a '***V***' suffix, such as ***nLength*** and ***nLengthV***. The variations are usually positive numbers indicating the magnitude of variation about the previous parameter. However, since a few special trees, like palms, require exceptions to common trends [REIL91], some parameters use the negative sign as a flag to activate a special mode. All angular parameters are specified in degrees. Likewise, angles in the equations are in degrees, unless otherwise stated. Except where noted, our equations describe structures based on our physical observations and research in tree reference manuals (see References).

Additionally, four trees parameter lists are given for comparison in Appendix. These specifications were designed using photographs in tree reference manuals. These trees, Quaking Aspen, Black Tupelo, Weeping Willow, and California Black Oak, can be seen in Plates 1q, 2, 5, and 1a, respectively. As trees vary widely and can be hard to identify even by experts [SYMO58], these specific definitions could be used to represent many different species of trees. Figure 1 is a diagram demonstrating some of the parameters. It does not show a complete tree, but rather exaggerates certain components to clarify their construction.

4.1 The Curved Stem

Our model is based on two elements, the stem and the leaf. We will use the generic term "stem" to refer to the trunk or branches at any level of recursion. The unit stem is a narrow near-conical tube whose relative z-axis is coincident with its central axis. Note that each stem has it own relative coordinate system. For a main branch whose z-axis points out perpendicularly to the trunk's z-axis, the branch's y-axis points up toward the sky and its x-axis points parallel to the ground surface, according to the right-hand rule. The tube of a stem at a recursive level ***n*** is divided into a number of near-cylindrical segments defined by ***nCurveRes***. Each segment is stored as a nearly-circular cross-section. These cross-sections are later connected together to draw a triangular mesh. If ***nCurveBack*** is zero, the z-axis of each segment on the stem is rotated away from z-axis of the previous segment by (***nCurve/nCurveRes***) degrees about its x-axis. If ***nCurveBack*** is non-zero, each of the segments in the first half of the stem is rotated (***nCurve***/(***nCurveRes/2***)) degrees and each in the second half is rotated (***nCurveBack***/(***nCurveRes/2***)) degrees. This two part curve allows for simple S-shaped stems. In either case, a random rotation of magnitude (***nCurveV***/***nCurveRes***) is also added for each segment. A special mode is used when ***nCurveV*** is negative. In that case, the stem is formed as a helix. The declination angle is specified by the magnitude of ***nCurveVary***.

4.2 Stem Splits

A stem generally extends out to the periphery of the tree, potentially splitting off cloned stems along its length. A cloned stem is considered at the same recursive level as its twin and inherits all of its properties. The frequency of splitting is defined by ***nSegSplits***. This is the number of new clones added for each segment along the stem and is usually between 0 and 1, with 1 referring to a dichotomous split on every segment. A value of 2

would indicate a ternary split. There is no pre-determined limit to the number of splits per segment; but, since each clone can also generate its own clones at the next segment, the resulting number of stems can easily reach undesirable levels. For instance, with a ***nCurveRes*** of 5 and ***nSegSplits*** of 2, one stem will eventually split off into 81 separate clones: $(\textbf{\textit{nSegSplits}}+1)^{\textbf{\textit{nCurveRes}}-1}=3^4$. Note in the top center diagram in Figure 1 where a tree has ***0SegSplits*** of 1 and ***0CurveRes*** of 3. The resulting splitting results in a trunk with four total stems: $(\mathbf{1}+1)^{3-1} = 4$. There is an additional parameter ***nBaseSplits*** that specifies the equivalent of ***nSegSplits*** at the end of the first segment of the trunk. This allows for an independent number of splits at the base of the tree, thus permitting trees that seem to have multiple trunks with few further splitting tendencies. Fractional values of ***nSegSplits*** will cause additional splits to be evenly distributed throughout all segments of all stems in that particular level of recursion. For example, an ***nSegSplits*** of 1.2 will form one clone on 80% of the level ***n*** segments and two clones on 20% of the segments. Note that this yields an average number of 1.2 splits per segment. Using random numbers simplistically to distribute the fractional part of ***nSegSplits*** is unacceptable because when, by chance, several consecutive segments all get the extra split, they can form an unnaturally large number of stems in close proximity on part of the tree. To evenly distribute the splits, we use a technique similar to Floyd-Steinburg Error Diffusion [FLOY76]. For each recursive level, a global value holds an "error value" initialized to 0.0. Each time ***nSegSplits*** is used, this error is added to create a $\text{SegSplits}_{\text{effective}}$ which is rounded to the nearest integer. The difference ($\text{SegSplits}_{\text{effective}}$-***nSegSplits***) is subtracted from the error. So, if a value is rounded up, it is more likely that the next value will be rounded down (and vice versa).

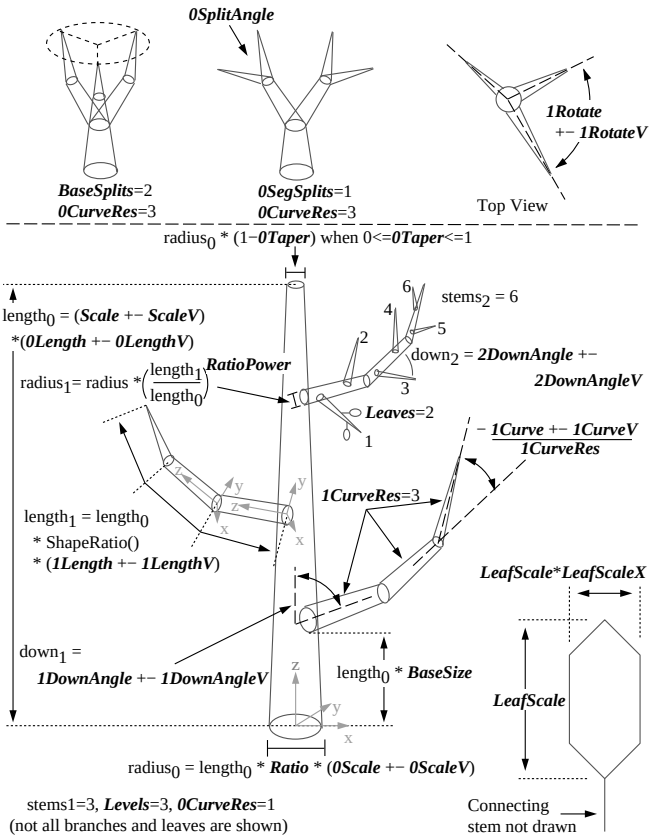

Figure 1: Tree Diagram

If there are any clones, then the z-axes of the stem and its clones each rotate away from the z-axis of the previous segment by

$$\text{angle}_{\text{split}} = (\textbf{\textit{nSplitAngle}}\pm\textbf{\textit{nSplitAngleV}}) - \text{declination}$$

limited to a minimum of 0, where the "declination" angle (defined here as the angle of a stem from the tree's positive z-axis) can be found by taking the inverse cosine of the z component of a unit z vector passed through the current matrix transformation of the relative coordinate system. The first clone continues the original mesh and cannot rotate around the z-axis or it would twist the mesh (i.e., if one rotated one of the circular faces on a cylinder about the longitudinal axis, the resulting section of geometry would render as a hourglass shape). This $\text{angle}_{\text{split}}$ is later distributed over the remaining segments in the reverse direction so that the stem will tend to return to its originally intended direction. This compensation prevents overspreading due to large numbers of stem splits. The extent that any level of stems spreads out can be easily controlled using the curve parameters.

A stem and its clones are also spread apart by rotating them about an axis that is parallel to the z-axis of the tree. This parallel axis of rotation intersects at the point where the split occurred. Note that they are not rotated about the relative z-axis of the stem as this would disturb the proper orientation of the relative x and y axes. In the normal case of a single clone, the original stem (which is continued after its clone is created) is rotated about the parallel axis by an angle of magnitude:

$$\left[\ 20 + 0.75 * (30 + \left|\ \text{declination}-90\ \right|) * \text{RANDOM}_{0\text{ to }1}{}^{2}\ \right]$$

The sign of this angle is random as well. This equation diverges two nearly-horizontal branches by 20 to 50 degrees about the parallel axis, but allows near-vertical branches to spread up to 140 degrees. Excessive rotation for a near-horizontal branch could cause a very unnatural effect.

4.3 Stem Children

One could theoretically build a tree just from clones, but the variety of trees you could produce would be greatly limited. The even distribution that makes the splits controllable also makes the shape formed from the resulting stems and clones very uniform. Also, many trees do not exhibit a clear splitting nature and have branches that grow from other branches in a spiral or nearly coplanar manner. For this, we can spawn children, which are considered one recursive level below their parents. Although a child can have entirely different attributes from its parent, some of these attributes are defined relative to its parent's equivalents. Note that the other models generally only allow each tree to be dichotomous, monopodial, or somewhere in between. Honda recognized a problem with excessive branching and sought to resolve it with branch interactions and unequal flow rates [HOND81]. Since our clones and children allow for dichotomous and monopodial branching simultaneously, we rarely encounter this problem. Also, since our parameters can address the character of an entire stem and not just its segment-to-segment nature, we allow users to make changes on a level they can more easily understand and visualize. ***nBranches*** defines the maximum number of child sub-stems that a particular level of stems can create over the length of all of its segments. The actual number of children from any stem might be less than this maximum. The number of successive child stems (really "grandchildren") is computed as

$$\text{stems} = \text{stems}_{\text{max}} * (0.2 + 0.8 * (\text{length}_{\text{child}}/\text{length}_{\text{parent}}) / \text{length}_{\text{child,max}})$$

for the first level of branches, and

$$\text{stems} = \text{stems}_{\text{max}} * (\ 1.0 - 0.5 * \text{offset}_{\text{child}}/\text{length}_{\text{parent}})$$

for further levels of branches, where $\text{offset}_{\text{child}}$ is the position in meters of the child along the parent's length (from the base). Any stem that has been cloned or is, itself, a clone reduces its propensity to form clones by half. Given a normalized position "ratio" from 0.0 to 1.0, a function ShapeRatio(shape, ratio) uses various pre-defined relations:

Shape	Result	
0 (conical)	0.2 + 0.8 * ratio	
1 (spherical)	0.2 + 0.8 * sin(π * ratio)	
2 (hemispherical)	0.2 + 0.8 * sin(0.5 * π * ratio)	
3 (cylindrical)	1.0	
4 (tapered cylindrical)	0.5 + 0.5 * ratio	
5 (flame)	ratio/0.7	ratio≤0.7
	(1.0 - ratio)/0.3	ratio>0.7
6 (inverse conical)	1.0 - 0.8 * ratio	
7 (tend flame)	0.5 + 0.5 * ratio/0.7	ratio≤0.7
	0.5 + 0.5 * (1.0 - ratio)/0.3	ratio>0.7
8 (envelope)	*use pruning envelope* (see Section 4.6)	

Generally, the ***Shape*** parameter is used as the index to this table of curves. These shapes correspond to generic shapes defined in the botanical tree texts, previously referenced.

The maximum relative length ($\text{length}_{\text{child,max}}$) of any recursive level of stems is ***nLength***±***nLengthV*** which is defined as a fraction of its parent's specific length. For example, a child with $\text{length}_{\text{child,max}}$ of 0.3 and a 10 meter long parent could reach a maximum length of about 3 meters. A length is computed by

$$\text{length}_{\text{child}} = \text{length}_{\text{trunk}} * \text{length}_{\text{child,max}} * \text{ShapeRatio}(\ (\textbf{\textit{Shape}}, (\text{length}_{\text{trunk}}-\text{offset}_{\text{child}})/(\text{length}_{\text{trunk}}-\text{length}_{\text{base}}))$$

for the first level of branches and

$$length_{child} = length_{child,max} * (length_{parent} - 0.6 * offset_{child})$$

for further levels of branches, where $length_{base}$ is the fractional bare area at the base of the tree calculated as (***BaseSize***$*scale_{tree}$) and $scale_{tree}$ defined as (***Scale***±***ScaleV***) in meters . The trunk has no parent, so its length is defined by

$$length_{trunk} = (\textbf{\textit{0Length}} \pm \textbf{\textit{0LengthV}})*scale_{tree}$$

If ***nDownAngleV*** is positive, the z-axis of a child rotates away from the z-axis of its parent about the x-axis at an angle of (***nDownAngle***±***nDownAngleV***). However, if ***nDownAngleV*** is negative, the variation is distributed along the height of the tree by

$$downangle_{child} = \textbf{\textit{nDownAngle}} \pm [\ \textbf{\textit{nDownAngleV}} * (1 - 2 * ShapeRatio(0, (length_{parent} - offset_{child}) / (length_{parent} - length_{base})))\ ]$$

This can be used to linearly change the down angle based on the position of the child along its parent, as with the Black Tupelo's main branches seen in Plate 2b. Note how they are angled upward near the crown of the tree and angled downward near the bottom. If ***nRotate*** is positive, each child formed along the parent is placed in a helical distribution by rotating about the z axis of its parent relative to the previous child by the angle (***nRotate***±***nRotateV***). In the special case where ***nRotate*** is negative, each child is rotated about its parent's z-axis relative to its parent's y-axis by the angle (180+***nRotate***±***nRotateV***) on alternating sides of the parent branch. This allows for a nearly coplanar child stem distribution. Since the y-axis of any stem with a small downangle points back toward its parent, the planar distribution from such a stem is aligned with that parent. This makes it easy to design trees where sub-branches tend to spawn parallel to the ground surface. This effect is most obvious in the tree shown in Plate 1v.

Aono and Kunii go into detail about the proper divergence and branching angle [AONO84]. These correspond to our rotation and down angles, respectively. They note the Schimper-Braun law which states that this divergence angle is a fraction of 360 degrees based on a Fibonacci sequence of 1/2, 1/3, 2/5, 3/8, ... , resulting in possible angles of 180, 120, 144, 135, and so on. Our results show that any number near 140 degrees works well in most situations. Aono and Kunii also note that the branching angle (our down angle) appears to be smaller for branches that form later as the tree matures. De Reffye attributes this to gravity affecting the increased mass of older branches and simulates the effect including elastic curvature using Young's modulus [REFF88]. The change in the branching angle can result in large angles at the base of the tree and smaller angles along the height of the tree. We implement this linearly with the negative ***nDownAngleV*** as noted above. However, Aono and Kunii state that changing their model to implement this effect does not add much realism. We find the effect, as implemented in our model, to be very substantial, especially in dormant or sparsely foliated trees.

4.4 Stem Radius

For all levels except the trunk, the radius at the base of a stem is defined as a function of the radius of its parent stem. The trunk's radius is proportional to the scale of the entire tree.

$$radius_{trunk} = length_{trunk} * \textbf{\textit{Ratio}} * \textbf{\textit{0Scale}} \quad \text{trunk}$$
$$radius_{child} = radius_{parent} * (length_{child} / length_{parent})^{\textbf{\textit{RatioPower}}} \quad \text{branches}$$

The maximum radius of a stem is explicitly limited to the radius of the parent at the point from which it was spawned. The radius of the stem can be tapered along its length. In the simplest form, this can be used to render the stem as a bent cone. However, there are other variations that allow for other cases according to the following chart:

nTaper	Effect
0	Non-tapering cylinder
1	Taper to a point (cone)
2	Taper to a spherical end
3	Periodic tapering (concatenated spheres)

Any fractional value from 0 to 3 is permitted to allow adjustment for a desired effect. The periodic tapering can be seen in the cactus of Plate 1(L) which has an ***0Taper*** of 2.2. For a normalized position Z from 0 to 1 along the length of a stem, the following equations compute $radius_Z$, the tapered radius in meters:

$$unit\_taper = \textbf{\textit{nTaper}} \qquad 0 \le \textbf{\textit{nTaper}} < 1$$
$$unit\_taper = 2 - \textbf{\textit{nTaper}} \qquad 1 \le \textbf{\textit{nTaper}} < 2$$
$$unit\_taper = 0 \qquad 2 \le \textbf{\textit{nTaper}} < 3$$

$$taper_Z = radius_{stem} * (1 - unit\_taper * Z) \quad \text{(purely tapered radius)}$$

and when $0 \le$ ***nTaper*** < 1

$$radius_Z = taper_Z$$

or when $1 \le$ ***nTaper*** ≤ 3

$$Z_2 = (1 - Z) * length_{stem}$$

$$depth = 1 \qquad (\textbf{\textit{nTaper}} < 2) \text{ OR } (Z_2 < taper_Z)$$
$$depth = \textbf{\textit{nTAPER}} - 2 \qquad \text{otherwise}$$

$$Z_3 = Z_2 \qquad \textbf{\textit{nTaper}} < 2$$
$$Z_3 = \left| Z_2 - 2 * taper_Z * integer(Z_2 / (2 * taper_Z) + 0.5) \right| \qquad \text{otherwise}$$

$$radius_Z = taper_Z \qquad (\textbf{\textit{nTaper}}<2) \text{ AND } (Z_3 \ge taper_Z)$$
$$radius_Z = (1-depth) * taper_Z + depth * sqrt(taper_Z^2 - (Z_3 - taper_Z)^2) \qquad \text{otherwise}$$

where 'depth' is a scaling factor used for the periodic tapering. This periodic tapering is useful for some cacti, where annual growth can accumulate in segments [HAUS91]. Similarly, it can be used as a rough approximation of the scales on palm trees.

In addition to tapering, the trunk may also vary its radius by other means. Flaring creates an exponential expansion near the base of the trunk. At the unit position Z from 0 to 1 along the length of a stem, the following $flare_Z$ scales the $radius_Z$ computed above.

$$y = 1 - 8 * Z$$
$$flare_Z = \textbf{\textit{Flare}} * (100^y - 1) / 100 + 1$$

where the value of y is limited to a minimum of zero. Note that this equation scales the radius by a minimum of 1 and a maximum of about (1 + ***Flare***).

The trunk can also have an irregular non-circular cross-section. This can be very apparent in the large supporting "knees" of cypress trees [REIL91]. These variations are also clearly present on some cacti, which can have pronounced ribs or ridges [HAUS91]. ***Lobes*** specifies the number of peaks in the radial distance about the perimeter. Even numbers can cause obvious symmetry, so odd numbers such as 3, 5, and 7 are preferred. The ***LobeDepth*** specifies the magnitude of the variations as a fraction of the radius as follows:

$$lobe_Z = 1.0 + \textbf{\textit{LobeDepth}} * \sin (\textbf{\textit{Lobes}} * angle)$$

given a specific "angle" from the x-axis about the z-axis. Note that a ***LobeDepth*** of zero effectively turns lobing off. The $lobe_Z$ value cumulatively scales $radius_Z$ as did $flare_Z$. Finally, a simple scaling factor (***0Scale***±***0ScaleV***) can also be applied to the trunk.

Bloomenthal modeled this flaring and lobing using an "equipotential curve surrounding the points of intersection of the tree skeleton with the plane of the contour" [BLOO85], essentially the blended circumference of several circles moving further away from the center of the trunk near the base of the tree.

4.5 Leaves

The recursive proliferation of children is limited by ***Levels***. This specifies the maximum level of stems that will be created starting from 0 for the trunk, usually to 3 or 4. If ***Leaves*** is non-zero, then leaves are used as the last level of recursion. The leaves use the ***nDownAngle***, ***nDownAngleV***, ***nRotate***, and ***nRotateV*** from the that level of recursion. Any leaves or stems beyond level 3 will simply use the parameters of level 3. Our most common configuration is to set ***Levels*** to 3 and ***Leaves*** to a non-zero value which would give you the following levels of recursion: trunk (0), branches (1), sub-branches (2), and leaves (3). Some trees, such as the weeping willow, require sub-sub-branches as well. ***Leaves*** specifies the density in the same manner as ***nBranches*** did for stems. As with stems, the actual density used is also dependent upon other factors such as the length of the parent branch relative to the maximum length for the parent's level. Given that the leaves are at the second level of recursion or further, the following density is used:

$$leaves\_per\_branch = \textbf{\textit{Leaves}} * ShapeRatio(4 \textit{ (tapered)}, offset_{child}/length_{parent}) * quality$$

given a quality factor supplied by the parent program that is usually near 1. This quality factor is also used to scale the leaves to maintain consistent coverage. This distribution of leaves has the natural effect of preferentially placing leaves near the outside of the tree. For a negative value of ***Leaves***, a special mode is used in which the leaves are placed in a fan at the end of the parent stem, as with some palm fronds. The angle over which the leaves fan out is specified by ***nRotate***. Note that when in fan mode, ***nRotate*** is not needed for its original purpose. A negative value can also

be applied to ***nBranches*** with similar results, but we have not yet modeled any trees requiring this attribute. We realize that these negative flags can become a bit confusing, but they are only used in a few special cases. Many users will never need them.

Leaves can assume many different shapes [CHND88]. We allow for a few common geometries of leaves based on ***LeafShape***. This parameter is used as an index to a list of pre-defined leaf shapes, such as oval, triangle, 3-lobe oak, 3-lobe maple, 5-lobe maple, any 3 leaflets. Each shape can be sized and scaled. For optimum coverage versus computational expense, the oval leaves are most commonly used. The pre-defined leaf geometries are stored with unit width and length. They are scaled as they are used to create the tree geometry. The length of the leaves, in meters, is determined by [***LeafScale*** / sqrt(quality)]. The width, in meters, is determined by [***LeafScale*****LeafScaleX*** / sqrt(quality)].

4.6 Pruning

Pruning is used to force a tree to fit inside a specific envelope. We originally avoided such a feature since we felt that the shape of a tree should proceed from its underlying structure, not from the use of artificial boundaries. We now concede that under the influence of certain environmental conditions or to control an "uncooperative" tree, pruning can be very useful. Prusinkiewicz demonstrates pruning applied to the L-system model [PRUS94]. Essentially, growth of branches is blocked by the edge of the envelope boundary. Since the L-system progressively grows connecting nodes, the model can simply hinder growth near a boundary. Our model must approach the problem differently. Since our stems often reach from the trunk to the tree's outer edges simply chopping off the ends of the offending branches,will make the tree's appearance suffer. While this may be the effect from some actual physical pruning, we would rather use pruning as a tool to influence the shape of a tree through the underlying structure. To do this, every stem must adjust its length to fit inside the envelope. Each stem must "know this new length" before it spawns any children since each child's length is dependent on its parent's length. Generally, the child branches are recursively spawned during the formation of the parent's segment from which they grow. This is necessary since the child must use a geometric transformation relative to the transformation of that segment. At that point, the ultimate extent of the parent is not known, so there must first be a non-recursive probing pass for each stem to measure and rescale its length and then a fully-recursive second pass to actually form the geometry and spawn the children. Note that the probing must also that each of the stem's clones fits. If any stem or clone punctures the boundary, its length can be iteratively reduced and re-probed until it fits.

While the model is capable of using any arbitrary envelope such as the topiary dinosaur in Prusinkiewicz's paper, the general user should be more comfortable with an easily selected simple envelope. Figure 2 shows a pruning envelope.

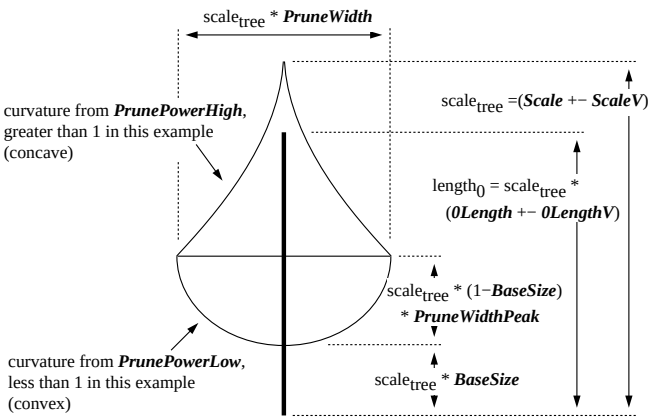

Figure 2: Pruning Diagram

The envelope covers a pseudo-ellipsoidal shape with a top at $scale_{tree}$ and bottom at (***BaseSize****$scale_{tree}$) in meters. The maximum width of the envelope is (***PruneWidth****$scale_{tree}$) in meters. This maximum width occurs at a position along the tree's z-axis as specified by ***PruneWidthPeak***. This peak is defined as the distance from the bottom of the envelope as a fraction of the total height of the envelope. A ***PruneWidthPeak*** of 0.5 would center the peak as with a standard ellipsoid. The curvature of the envelope can also be independently controlled above and below the peak using ***PrunePowerHigh*** and ***PrunePowerLow***, respectively. A power of 1 gives a linear envelope from ***PruneWidth*** to 0 over the distance from ***PruneWidthPeak*** to the top or bottom of the envelope. A power of 2 gives a rounded concave envelope, while a power of 0.5 gives a rounded convex envelope. To determine whether a given transformed point (x,y,z) is inside the envelope, the boolean 'inside' is computed as:

$$r = \text{sqrt}\ (x^2 + y^2 + z^2)$$

$$\text{ratio} = (scale_{tree} - z) / (scale_{tree} * (1 - \textbf{\textit{BaseSize}}))$$

$$\text{inside} = [\ r / scale_{tree} < \textbf{\textit{PruneWidth}} * \text{ShapeRatio}(8\ \textit{(envelope)},\ \text{ratio})\ ]$$

where ShapeRatio(8, ratio) is defined as

$$[\ \text{ratio} / (1 - \textbf{\textit{PruneWidthPeak}})\ ]^{\textbf{\textit{PrunePowerHigh}}}$$

when ratio < 1 - ***PruneWidthPeak***, or

$$[\ (1 - \text{ratio}) / (1 - \textbf{\textit{PruneWidthPeak}})\ ]^{\textbf{\textit{PrunePowerLow}}}$$

when ratio ≥ 1 - ***PruneWidthPeak***. ShapeRatio(8, ratio) always returns 0 when ratio is not in the range of 0 to 1. The ***Shape*** parameter can use this index of 8 for a custom shape even if pruning is not turned on. This allows the user to define a shape not covered by the predefined shapes. If ***Shape*** is 8 and pruning is on, the tree will tend to match the customized shape even before pruning takes place. This may cause some strongly curved stems to fall short of the envelope's edge.

The effects of pruning can be diminished by using the ***PruneRatio***. This defines a weighted average between the unpruned original length and the completely pruned length. A ***PruneRatio*** of 1 activates full pruning while a ***PruneRatio*** of 0.0 effectively turns pruning off. Thus with values between 0 and 1, partial pruning can be utilized to avoid artificially smooth boundaries. Plate 7 demonstrates the use of the pruning envelope to control the weeping willow.

4.7 Wind Sway

We model stem bending as the deflection of an elastic rod with a circular cross-section fixed on one end. This rod has a uniformly distributed force applied to it. The solution of this is a classic problem of mechanics where applying the Myosotis method yields useful solutions for the deflection [HART77]. We then consider this rod as a kind of pendulum [HALL88]. The entire system is then modeled as the superposition of coupled oscillators whose periods and phase angles differ so that the paths of points on a stem are very complex Lissajous figures [ALON70]. These results confirm our general observation that light to moderate winds induce trees to move so that branches sway at various directions and rates of oscillations. We currently model the oscillatory effects observed for light to moderate winds only.

In our model, tree movement is simulated by introducing time-variant curvature changes to the stem segments. This effect is added to the structural curvature introduced by ***nCurve*** and ***nCurveBack*** causing rotations between segments about both the x and y axes. With wind speeds varying from $wind_{speed}$ to ($wind_{speed}$+ $wind_{gust}$), the sway angles $sway_x$ and $sway_y$ at unit position Z from 0 to 1 of a segment along the length of a stem are computed at any "time" (in seconds) using:

$$a_0 = 4 * length_{stem} (1 - Z) / radius_Z \quad \text{(degrees)}$$

$$a_1 = wind_{speed} / 50 * a_0 \quad \text{(degrees)}$$

$$a_2 = wind_{gust} / 50 * a_0 + a_1/2 \quad \text{(degrees)}$$

$$b_x = sway\_offset_x + radius_{stem} / length_{stem} * time/15 \quad \text{(radians)}$$

$$b_y = sway\_offset_y + radius_{stem} / length_{stem} * time/15 \quad \text{(radians)}$$

$$sway_x = [\ a_1 * \sin(b_x) + a_2 * \sin (0.7 * b_x)\ ] / \textbf{\textit{nCurveRes}} \quad \text{(degrees)}$$

$$sway_y = [\ a_1 * \sin(b_y) + a_2 * \sin (0.7 * b_y)\ ] / \textbf{\textit{nCurveRes}} \quad \text{(degrees)}$$

The angles $sway\_offset_x$ and $sway\_offset_y$ are randomly selected for each stem in the tree. When the wind sway is activated, each tree geometry description must be reformed for each frame in an animation to adapt to the new angles. By using the same random seed, a specific tree will always have the same basic geometry, perturbed only by the wind-activated curvature variations. The angles $sway_x$ and $sway_y$ cause rotations between segments about the x and y axes, respectively.

4.8 Vertical Attraction

With even hemispherical illumination (sky shine), tree shoots grow upwards because they are negatively geotropic and positively phototropic. An upward growth tendency is usually a subtle effect and can be implemented using the declination and orientation of each segment in each stem. For sub-branches and beyond, this curving effect is used in addition to the other curvature effects. The trunk and main branches do not use these functions

since any such effect can be more easily controlled through the previous curve parameters. The ***AttractionUp*** parameter specifies the upward tendency. Zero denotes no effect and negative numbers cause downward drooping as in the Weeping Willow. A magnitude of one results in a tendency of each stem to curve just enough so that its last segment points in a vertical direction. Higher magnitudes cause stems to curve toward the vertical much sooner. Very high magnitudes such as 10 may result in snaking oscillations due to over-correction. This is not necessarily an undesirable result since branches on some trees exhibit a distinctly sinusoidal shape characteristic. Once the effects of ***nCurve*** are introduced to a segment, $curve\_up_{segment}$ is computed for each segment as

$$declination = \cos^{-1}(transform\_z_z) \quad \text{(radians)}$$
$$orientation = \cos^{-1}(transform\_y_z) \quad \text{(radians)}$$

$$curve\_up_{segment} = \textbf{\textit{AttractionUp}} * declination * \cos (orientation) / \textbf{\textit{nCurveRes}} \quad \text{(radians)}$$

where $transform\_z_z$ is the z component of a unit z vector passed through the current viewing transformation and $transform\_y_z$ is the z component of a unit y vector passed through the current viewing transformation. This $curve\_up_{segment}$ is added to the segment's curvature.

4.9 Leaf Orientation

Left alone, the modeled leaves will generally assume seemingly random orientations. However, in reality, leaves are oriented to face upwards and outwards, presumably to optimize the available direct (sun) and scattered (sky) light. We can use the declination and orientation of each leaf to rotate them toward the light. The necessary rotations are computed based on the current viewing transformation and are applied prior to creating the leaf into the geometric description. The effect, fractionally controlled by "bend", is applied by obtaining the leaf's position ($leaf_x$, $leaf_y$, $leaf_z$) and normal ($leaf_{nx}$, $leaf_{ny}$, $leaf_{nz}$) in tree coordinates from the current transform matrix, then computing the current and desired angles:

$$theta_{position} = atan_2 (leaf_y, leaf_x)$$
$$theta_{bend} = theta_{position} - atan_2 (leaf_{ny}, leaf_{nx})$$

then computing the change:

$$rotate_z (bend * theta_{bend})$$

then recomputing declination, orientation, and normal vector using new transform:

$$phi_{bend} = atan_2 (sqrt(leaf_{nx}^2 + leaf_{ny}^2), leaf_{nz})$$
$$rotate_z (-orientation)$$
$$rotate_x (bend * phi_{bend})$$
$$rotate_z (orientation)$$

Plate 6 shows the bending effect applied to a Sassafras tree. The modified leaf orientations greatly increase the diffusive reflections from the tree. The increased variations improve the overall appearance.

5 DEGRADATION AT RANGE

A tree generated with our algorithm may have on the order of 5000 to 100,000 facets. The detail can be increased automatically for even higher resolution images, such as the Weeping Willow in Plate 3 boosted to over 1 million facets. Currently, a high-end graphics workstation may be capable of only about 50,000 facets in real time. The high resolution of the trees is necessary to have an accurate representation at close ranges of 10 to 50 meters or in equivalent magnified views of greater ranges, as in narrow fields-of-view. However, at long ranges, such as 1000 meters, a much lower resolution tree could be rendered faster with little or no loss in apparent quality.

At first thought, it may seem useful to form multiple geometric descriptions of the same tree at different "levels of detail". At longer ranges, progressively lower resolution geometric descriptions would be used. This approach has two problems. First, each instance of a tree consumes resources. An average tree's geometric description may use about 1Mb of RAM. Also, it may require 1 to 10 seconds to form the data. These numbers become much more significant when multiplied by, perhaps, 100 instances. While this could be managed, a more critical second problem arises with the quantization of the resolution. In a still picture, the changes between resolutions would not be very apparent since the variably resolved trees appear as different trees. However, in a dynamic simulation, specific trees would switch from one resolution to the next. This would result in wide "resolution waves" flowing through forest canopies. This is unacceptable for realistic simulation.

A method is needed that uses a single geometric description and renders it at an optimal resolution for any range. The changes between the differently resolved geometries must be very fine, preferably corresponding with removal or modification of each facet one at a time. There should be negligible overhead (CPU and RAM) involved with this reinterpretation of the specified geometry.

Since the trees are not arbitrary objects, we can fit a range-degradation algorithm to their expected geometry. Each tree geometry is organized into four discrete geometric descriptions: 3 stem levels and the leaves. Any stems beyond the third level are grouped with the third level. The deeper levels of stems are rarely visible at long ranges and are often obscured by the leaves. Oppenheimer recognized that he could use polygonal tubes for large-scale details and vectors (lines) for the smaller details [OPPE86]. He warns that artifacts can occur if the "cutover" level is not deep enough. He also states that many small branches can be rendered as triangular tubes. Our method of rendering makes similar approximations for efficiency.

To most efficiently use the CPU and memory, our technique does not convert the geometry, it merely re-interprets it. With progressively increasing ranges, a tree will re-interpret stem meshes as lines and leaf polygons as points. With longer ranges, some individual stems and leaves will disappear altogether. The specific geometry at any range can be rendered properly by altering limits and increments in the loops that draw the data. Although we speak of removing items one by one, we do not actually mark or delete them. We merely change the loop parameters that scan the stored geometry so that items are skipped. Any number of arbitrarily-ranged trees can be drawn in any order. The time and space overhead required to compute and hold these boundary limits is negligible. A 100,000 facet tree geometry may be rendered at 2 kilometers as about 30 lines and 1000 points. This allows vast expanses of trees to be drawn very quickly. A viewer can then move close to any of these trees and see them at their full resolution.

Since the items in each geometric description are ordered in the same manner as they were created, they generally start from the bottom of the tree and work up. The items are not randomly organized; therefore, we cannot simply remove objects one at a time in order from the top or bottom of the list. This could cause the top of the tree to be heavily degraded while the bottom remained unchanged, or vice versa. Instead, we group the items of a type of geometry into groups of a small size which we will call "masses". The number of elements per mass is determined by an appropriate "mass_size". We use a mass_size of 16 for all the stems and 4 for the leaves. Curve fitting equations give a value between 0 and mass_size. To explain, we will use the general term "primitive" to refer non-specifically to polygons, lines, or points and the general term "item" to refer non-specifically to leaves, trunk, branches, or sub-branches. The total number of elements in the geometric description of any item is given as "$total\_number_{item}$". Of this, we wish to draw a certain fraction of these items using a specific primitive. The portion to be drawn is specified by the non-integer "$number_{primitive,item}$", which is between 0 and $mass\_size_{primitive,item}$. For example, a $mass\_size_{lines,1}$ of 16 divides up main branch lines into masses of 16. A $number_{lines,1}$ of 5 says that for every 16 cross-sections of recursion level 1, there should be lines connecting the first five. Fractional numbers will draw an additional item for a percentage of the masses. If there were 160 main-branch cross-sections (10 masses) and $number_{lines,1}$ of 5.3, then the first 3 masses would show 6 of 16 lines and the last 7 masses would show 5 of 16 lines. A loop to draw the reduced portion of the item using a specific primitive would be:

$int\_number_{primitive,item} = integer(number_{primitive,item})$
$masses_{primitive,item} = total\_number_{item} / mass\_size_{primitive,item}$
$change_{primitive,item} = masses_{primitive,item} * (number_{primitive,item} - int\_number_{primitive,item})$

```
for mass = 0 to masses_primitive,item
    {
    start = mass * mass_size_primitive,item
    end = start + int_number_primitive,item

    if mass < change_primitive,item
        end = end + 1

    for index = start to end
        draw_primitive,item( index )
    }
```

To compute the necessary $number_{primitive,item}$, we need to first convert the range to a calibrated scale. This adjusts for the current image size and vertical field of view. A modified range value, r_2, is computed as:

$$r_2 = range * 1000 / height_{image} * Field\_Of\_View_y / 60$$

This compensates for the effect of a telephoto lens that causes a tree to appear to be much closer.

The following equations outline how $number_{primitive,item}$ is computed for different levels at different ranges. First, we use the general quality factor supplied by the parent program (usually between 0 and 1) to determine some general scaling factors:

s = quality / 2	tree is evergreen, or in summer and fall
s = quality	otherwise
d = 100	in spring
d = 200	otherwise

Then, we compute the polygons, lines, and points needed for each display item as follows.

Level 0 Stems (trunk)

	r_2 <100	don't draw trunk lines (can appear as seam)
100 <	r_2	draw all trunk lines
	r_2 < 300	$number_{polygons,0} = mass\_size_{polygons,0}$
300 <	r_2 < 800	$number_{polygons,0} = mass\_size_{polygons,0} * [1.5 - r_2/600]$
800 <	r_2	don't draw trunk polygons at all

Level 1 Stems (main branches)

	r_2 < 200s	$number_{polygons,1} = mass\_size_{polygons,1} * [1.5 - r_2/600]$ (bounded 0 to $mass\_size_{polygons,1}$)
		$number_{lines,1} = mass\_size_{lines,1}$
200s <	r_2 < 2000s	don't draw polys
		$number_{lines,1} = mass\_size_{lines,1} * [2.2 - 1.2 (r_2/200s)^{0.3}]$
2000s <	r_2	draw nothing for main branches

Level 2 Stems (other branches)

	r_2 < 50s	$number_{polygons,2} = mass\_size_{polygons,2}$
		$number_{lines,2} = mass\_size_{lines,2}$
50s <	r_2 < 100s	$number_{polygons,2} = mass\_size_{polygons,2} * [2 - r_2/50s]$
		$number_{lines,2} = mass\_size_{lines,2}$
100s <	r_2 < 500s	don't draw polys
		$number_{lines,2} = mass\_size_{lines,2} * [2 - (r_2/100s)^{0.5}]$
500s <	r_2	draw nothing for secondary branches

Leaves

	r_2 < d/4	$number_{polygons,3} = mass\_size_{polys,3}$
		$number_{points,3} = mass\_size_{points,3}$
d/4	r_2 < d	$number_{polygons,3} = mass\_size_{polys,3} * [4/3 - r_2 / (3d/4)]$
		$number_{points,3} = mass\_size_{points,3}$
d <	r_2	$number_{polygons,3} = 0$
		$number_{points,3} = mass\_size_{points,3} * [1.5 - r_2 / 2d]$ (minimum of 1)

The effects of these equations can be seen in Table 2 which summarizes the total number of triangles, lines, and points. Triangles refer to elements of the triangular meshes which comprise the polygons.

Item	5m	30m	60m	120m	240m	600m	1200m
Level 0 Triangles	1440	1440	1440	1440	1440	760	0
Level 0 Lines	0	0	0	36	36	36	36
Level 1 Triangles	960	960	960	0	0	0	0
Level 1 Lines	240	240	240	223	153	35	0
Level 2 Triangles	17736	14580	0	0	0	0	0
Level 2 Lines	5912	5912	5363	2648	0	0	0
Leaf Triangles	53248	53248	49800	28200	0	0	0
Leaf Points	13312	13312	13312	13312	11944	1664	1664

Table 2: Number of elements drawn at specific ranges in summer on Quaking Aspen

Plate 4 shows the Quaking Aspen rendered at the ranges listed in Table 2, excluding 5 meters, progressively zoomed by powers of two.

6 APPLICATION

Our project involves the development of software to produce accurate and realistic high resolution imagery in both the visible and infrared spectrums. The emphasis is on positioning vehicles in the context of natural environments for studies of detection and recognition by both humans and machines. The backgrounds and vehicles must be of equally high fidelity to alleviate any bias in the testing.

The software utilizes readily available elevation maps and creates synthetically-generated shading variations for numerous natural effects. Readily available feature maps are usually at a poor resolution and often only describe trees as deciduous, coniferous, neither, or both. This is inadequate for our needs. We create our own feature maps from any available information such as scenario data (meteorological, topographical, vegetative type and placement, etc.), satellite imagery, and photographs, both aerial and at ground level. We describe vegetation using a 16-bit raster feature map where 14 bits specify 14 trees or related objects (not mutually-exclusive), and 2 bits specify 3 types of grass (mutually-exclusive). Any of the 14 trees (from a larger list) and 3 grass types can be selected differently for any specific scene. For any one of the 14 selected trees, any number of variations can be specified. These variations use the same parameter file, but are generated from a different random seed. During rendering, these variations of a tree type are spread randomly over positions where the appropriate bit for that tree is set in the vegetation feature map. This can prevent large forests of similar tree types from appearing too uniform and self-similar. Other feature maps are used for soils, rocks, waterways, and roadways. Currently, we generally resolve elevation and features maps at 1 or 2 meters per sample. The grass resembles that of Reeves and Blau's particle grass [REEV85]. In our case, the grass is drawn as curved lines composed of Gouraud-shaded segments.

Shadowing within the trees is produced using a standard shadow map technique [FOLE92]. The shadow map can be used to mark which geometric components (polygons, lines, points) will be shadowed before rendering takes place. This technique is only valid if each rendered instance of each specific geometric description has the same rotational orientation about its z-axis. Otherwise, the shadows would be rotated with the tree. This restriction is usually acceptable since each tree selected can have multiple variations, each of which can be randomly scaled for each instance.

Plate 3 shows various images of a visual simulation from different points of view. Plate 3f was made from an altered scenario with higher tree density. A moderate haze was applied to the image in Plate 3b.

7 CONCLUSION

We have introduced a model based on geometrical observables to create and render three-dimensional trees for simulating natural scenery. A wide variety of complex realistic tree structures can be generated quickly using a small number of parameters. The resulting images appear quite similar to images of real botanical trees. We have demonstrated the efficient use of the model with our synthetic scene generator.

We explained a range degradation methodology to smoothly degrade the tree geometry at long ranges. This is used to optimize the drawing of large quantities of trees in forested areas.

Our attention in designing the model was focused on allowing a general user to create trees that generally match images from books or photographs. The user needs no knowledge of botany or complex mathematical principles, only a basic understanding of geometry. We concentrated on the general structural appearance of a tree instead of the biological and biophysical principles that produced its structure.

Currently, the rendering of our trees at close range is not quite fast enough to meet the needs of real time simulation. A high end graphics workstation may only be able to draw one very close tree or a few dozen long range trees in real time. However, newer hardware will inevitably bring higher performance. In the near future, tree models such as this will be important in many areas of computer graphics.

ACKNOWLEDGEMENTS

All images were created with the CREATION software developed by Teletronics and the Modeling Simulation Branch of the US Army Research Laboratory.

Thanks to the US Army Research Lab who supported this project and helped with this paper, specifically Teresa Kipp, John Ho (retired), Gertrude Kornfeld (retired), Hung Nguyen, E. "Glenn" Dockery, Michael Lander, Janice Colby, and Giap Huynh, and also Dickson Fang and Scott Hawley of Teletronics. Thanks also to Dynamics Research Corporation.

REFERENCES

[ALON70] M. Alonso, E. Finn. *Physics.* Addison-Wesley, Reading, Massachusetts, 1970, pp. 160-166.

[AONO84] M. Aono, T. Kunii. Botanical Tree Image Generation. *IEEE Computer Graphics and Applications.* May, 1984, pp.10-34, Volume 4, No.5.

[BLOO85] J. Bloomenthal. Modeling the Mighty Maple. Proceedings of SIGGRAPH '85 (San Francisco, California, July 22-26, 1985). In *Computer Graphics* Proceedings, Annual Conference Series, 1985, ACM SIGGRAPH, pp. 305-311.

[CHND88] P. Chandler, A. Cook, G. DeWolf, G Jones, K Widin. *Taylor's Guide to Trees.* Houghton Mifflin Company, Boston, 1988.

[CHAN82] F. Chan, F. Ching, W Collins, M. Evans, W. Flemer, J. Ford, F. Galle, R Harris, R. Korbobo, F. Lang, F. Mackaness, B. Mulligan, R. Ticknor. *Trees.* The American Horticultural Society, Mount Vernon, Virginia, 1982.

[COLL74] G. Collingwood, W. Brush. *Knowing Your Trees.* The American Forestry Association, Washington, DC., 1974.

[FOLE92] J. Foley, A. vanDam, S. Feiner, J. Hughes. *Computer Graphics, Principles and Practice,* Second Edition. Addison-Wesley, Reading, Massachusetts, 1992.

[FLOY76] R. Floyd, L. Steinburg. An Adaptive Algorithm for Spatial Grey Scale, *Proceedings SID.* 1976, pp. 75-77.

[HALL88] D. Halliday, R, Resnick. *Fundamentals of Physics,* 3rd Ed. J. Wiley & Sons, New York, 1988, pp. 306-322.

[HAUS91] E. Haustein. *The Cactus Handbook.* Hamlin, London, 1991.

[HART77] J. Den Hartog. *Strength of Materials.* Dover, Mineola, 1977, pp. 79-88.

[HOND71] H. Honda. Description of the Form of Trees by the Parameters of the Tree-like Body: Effects of the Branching Angle and the Branch Length on the Shape of the Tree-like Body. *Journal of Theoretical Biology.* 1971, pp. 331-338.

[HOND81] H. Honda, P. Tomlinson, J. Fisher. Computer Simulation of Branch Interaction and Regulation by Unequal Flow Rates in Botanical Trees. *American Journal of Botany.* 1981, pp. 569-585.

[LIND68] A. Lindenmayer. Mathematical Models for Cellular Interactions in Development, I&II. *Journal of Theoretical Biology.* 1968, pp. 280-315.

[OPPE86] P. Oppenheimer. Real Time Design and Animation of Fractal Plants and Trees. Proceedings of SIGGRAPH '86 (Dallas, Texas, August 18-22, 1986). In *Computer Graphics* Proceedings, Annual Conference Series, 1986, ACM SIGGRAPH, pp. 55-64.

[PAGE93] J. Page. *Planet Earth: Forest.* Time-Life Books, Alexandria, Virginia, 1983.

[PRUS90] P. Prusinkiewicz, A. Lindenmayer. *The Algorithmic Beauty of Plants.* Springer-Verlag, New York, 1990.

[PRUS94] P. Prusinkiewicz, M. James, R. Měch. Synthetic Topiary. Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24-29, 1994). In *Computer Graphics* Proceedings, Annual Conference Series, 1994, ACM SIGGRAPH, pp. 351-358.

[REEV85] W. Reeves. Approximate and Probabilistic Algorithms for Shading and Rendering Structured Particle Systems. Proceedings of SIGGRAPH '85 (San Francisco, California, July 22-26, 1985). In *Computer Graphics* Proceedings, Annual Conference Series, 1985, ACM SIGGRAPH, pp. 313-322.

[REFF88] P. de Reffye, C. Edelin, J. Françon, M. Jaeger, C. Puech. Plant models faithful to botanical structure and development. Proceedings of SIGGRAPH 88 (Atlanta, Georgia, August 1-5, 1988). In *Computer Graphics* Proceedings, Annual Conference Series, 1988, ACM SIGGRAPH, pp. 151-158.

[REIL91] A. Reilly. *The Secrets of Trees.* Gallery Books, New York, 1991.

[ROHL94] J. Rohlf, J. Helman.. IRIS Performer: A High Performance Multiprocessing Toolkit for Real-Time 3D Graphics. Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24-29, 1994). In *Computer Graphics* Proceedings, Annual Conference Series, 1994, ACM SIGGRAPH, pp. 381-394, specifically Figures 14 and 17 on page 394.

[SYMO58] G. Symonds. *The Tree Identification Book.* William Morrow & Company, New York, 1958.

[TOOT84] E. Tootill, S. Blackmore. *The Facts on File Dictionary of Botany.* Market House Books LTD, Aylesbury, UK, 1984, p.155.

APPENDIX: Parameter List

Parameter	Description	Quaking Aspen	Black Tupelo	Weeping Willow	CA Black Oak
Shape	general tree shape id	7	4	3	2
BaseSize	fractional branchless area at tree base	0.4	0.2	0.05	0.05
Scale,ScaleV,ZScale,ZScaleV	size and scaling of tree	13, 3, 1, 0	23, 5, 1, 0	15, 5, 1, 0	10, 10, 1, 0
Levels	levels of recursion	3	4	4	3
Ratio,RatioPower	radius/length ratio, reduction	0.015, 1.2	0.015, 1.3	0.03, 2	0.018, 1.3
Lobes,LobeDepth	sinusoidal cross-section variation	5, 0.07	3, 0.1	9, 0.03	5, 0.1
Flare	exponential expansion at base of tree	0.6	1	0.75	1.2
0Scale,0ScaleV	extra trunk scaling	1, 0	1, 0	1, 0	1, 0
0Length,0LengthV, 0Taper	fractional trunk, cross-section scaling	1, 0, 1	1, 0, 1.1	0.8, 0, 1	1, 0, 0.95
0BaseSplits	stem splits at base of trunk	0	0	2	2
0SegSplits,0SplitAngle,0SplitAngleV	stems splits & angle per segment	0, 0, 0	0, 0, 0	0.1, 3, 0	0.4, 10, 0
0CurveRes,0Curve,0CurveBack,0CurveV	curvature resolution and angles	3, 0, 0, 20	10, 0, 0, 40	8, 0, 20, 120	8, 0, 0, 90
1DownAngle,1DownAngleV	main branch: angle from trunk	60, -50	60, -40	20, 10	30, -30
1Rotate,1RotateV,1Branches	spiraling angle, # of branches	140, 0, 50	140, 0, 50	-120, 30, 25	80, 0, 40
1Length,1LengthV,1Taper	relative length, cross-section scaling	0.3, 0, 1	0.3, 0.05, 1	0.5, 0.1, 1	0.8, 0.1, 1
1SegSplits,1SplitAngle,1SplitAngleV	stem splits per segment	0, 0, 0	0, 0, 0	0.2, 30, 10	0.2, 10, 10
1CurveRes,1Curve,1CurveBack,1CurveV	curvature resolution and angles	5, -40, 0, 50	10, 0, 0, 90	16, 40, 80, 90	10, 40, -70, 150
2DownAngle,2DownAngleV	secondary branch: angle from parent	45, 10	30, 10	30, 10	45, 10
2Rotate,2RotateV,2Branches	spiraling angle, # of branches	140, 0, 30	140, 0, 25	-120, 30, 10	140, 0, 120
2Length,2LengthV, 2Taper	relative length, cross-section scaling	0.6, 0, 1	0.6, 0.1, 1	1.5, 0, 1	0.2, 0.05, 1
2SegSplits,2SplitAngle,2SplitAngleV	stem splits per segment	0, 0, 0	0, 0, 0	0.2, 45, 20	0.1, 10, 10
2CurveRes,Curve,2CurveBack,2CurveV	curvature resolution and angles	3, -40, 0, 75	10, -10, 0, 150	12, 0, 0, 0	3, 0, 0, -30
3DownAngle,3DownAangleV	tertiary branch: angle from parent	45, 10	45, 10	20, 10	45, 10
3Rotate,3RotateV,3Branches	spiraling angle, # of branches	77, 0, 10	140, 0, 12	140, 0, 300	140, 0, 0
3Length,3LengthV, 3Taper	relative length, cross-section scaling	0, 0, 1	0.4, 0, 1	0.1, 0, 1	0.4, 0, 1
3SegSplits,3SplitAngle,3SplitAngleV	stem splits per segment	0, 0, 0	0, 0, 0	0, 0, 0	0, 0, 0
3CurveRes,3Curve,3CurveBack,3CurveV	curvature resolution and angles	1, 0, 0, 0	1, 0, 0, 0	1, 0, 0, 0	1, 0, 0, 0
Leaves,LeafShape	number of leaves per parent, shape id	25, 0	6, 0	15, 0	25, 0
LeafScale,LeafScaleX	leaf length, relative x scale	0.17, 1	0.3, 0.5	0.12, 0.2	0.12, 0.66
AttractionUp	upward growth tendency	0.5	0.5,	-3	0.8
PruneRatio	fractional effect of pruning	0	0	1	0
PruneWidth,PruneWidthPeak	width, position of envelope peak	0.5, 0.5	0.5, 0.5	0.4, 0.6	0.5, 0.5
PrunePowerLow,PrunePowerHigh	curvature of envelope	0.5, 0.5	0.5, 0.5	0.001, 0.5	0.5, 0.5

Plate 1: (a,b) Black Oak, (c,d) Sassafras, (e,f) Swamp Oak, (g) Cottonwood, (h) Tamarack, (i,j) Lombardy Poplar, (k) Queen Palm, (l) cactus, (m,n) Rock Elm, (o) Black Spruce, (p) Austrian Pine, (q,r) Quaking Aspen, (s) Balsam Fir, (t) White Cedar, (u) Bamboo, (v) *generic*, (w) Jack Pine, (x) Longleaf Pine

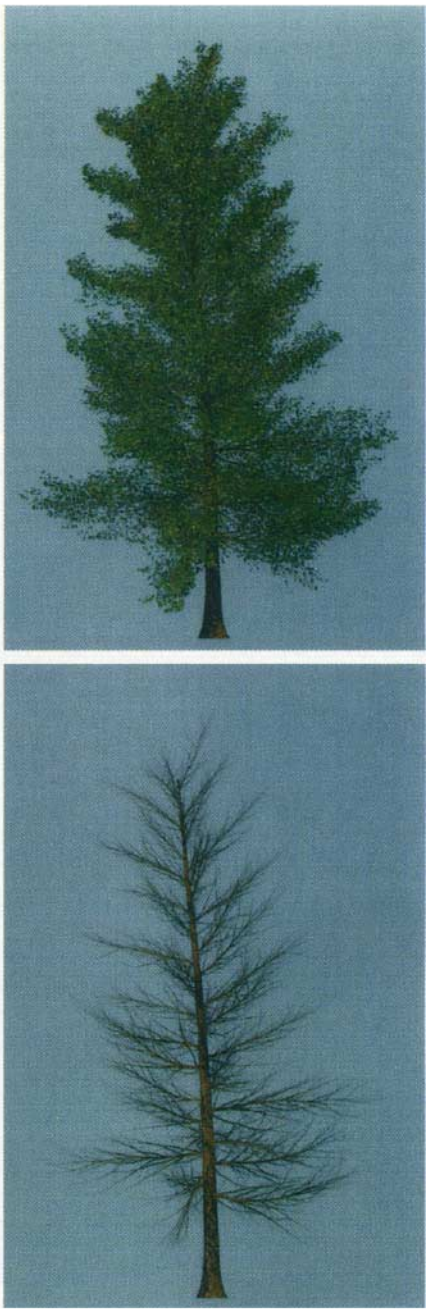

Plate 2: Black Tupelo,
(a) with leaves (b) without leaves

Plate 3: Scene from (a) Northeast, (b) Northwest, (c,d) South, (e) Overhead; and (f) with Modifications

Plate 4: Range Degradation

Plate 5: Weeping Willow

Plate 6: Sassafras with leaves (a) unmodified, (b) re-oriented

Plate 7: Weeping Willow: (a) unpruned, (b) pruned