EUROGRAPHICS Symposium on Sketch-Based Interfaces and Modeling (2012)
L. B. Kara and K. Singh (Editors)

TreeSketch: Interactive Procedural Modeling of Trees on a Tablet

Steven Longay[1], Adam Runions[1], Frédéric Boudon[2] and Przemyslaw Prusinkiewicz[1]

[1] University of Calgary, Canada
[2] CIRAD/INRIA, Virtual Plant Team, UMR AGAP, Montpellier, France

Figure 1: *Examples of trees created with TreeSketch. Arrows indicate the motions of the brush that determined the corresponding tree forms. The trees were generated instantaneously while brushing.*

Abstract

TreeSketch is a system for modeling complex trees that look natural yet are creatively designed. The system integrates procedural tree generation with a multi-touch tablet interface that provides detailed control of tree form. The procedural component is based on the concept of tree self-organization and simulates competition of branches for space and light as the tree develops from a seedling. The modeler can control this process by directing growth with a procedural brush, changing parameters as the tree grows, interleaving controlled and autonomous growth, and editing generated forms. Complex trees can be created in a matter of seconds.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.5]: Computational Geometry and Object Modeling—Geometric algorithms, languages, and systems; Computer Graphics [I.3.6]: Methodology and Techniques—Interaction techniques; Simulation and Modeling [I.6.8]: Types of Simulation—Visual.

1. Introduction

> *Our [modeling] tools need to feel more like brushes.*
> Rob Cook, 2009 Steven A. Coons Award speech

Due to the complexity of tree structures, the creation of trees for computer imagery requires specialized modeling methods [SLS*10]. Current methods include procedural tree generation and the reconstruction of tree geometry from photographs or laser scans. Reconstruction methods can potentially provide faithful models of real trees with little user intervention. However, finding an actual tree that meets the artistic requirements of a specific scene while being sufficiently isolated to obtain good photographs or scans is difficult. This problem is compounded for trees that only grow in forests, and accentuated by the limited geographic range in which some trees occur. Procedural models do not suffer from these limitations. Nevertheless, the development of

DOI: 10.2312/SBM/SBM12/107-120

Figure 2: *Designing a bonsai-inspired tree. The modeler has first applied three strokes of a small-sized procedural brush to create three superimposed arched branches (left). The design is completed by pruning the overhanging parts of the first two arcs, adding smaller branchlets with several strokes of a larger brush, increasing the girth of the main trunk, and stretching the trunk vertically (right).*

methods capable of generating widely diverse trees, botanically plausible yet easily controlled to meet artistic requirements, has remained a challenge.

In this paper we present TreeSketch, a program that allows the modeler to create a wide range of trees by combining procedural methods with detailed control of tree form using a multi-touch tablet interface. The main tool is a brush, which gives the modeler decisive control over the form of the tree (Figure 1). By changing the brush radius, the modeler can seamlessly progress from specifying the exact course of individual axes to a broad definition of large branches and the entire crown. The user can also direct the growth into predefined shapes using a lasso tool to draw silhouettes, or allow the tree to grow autonomously. Reversing time "ungrows" the tree, providing a continuous undo and making it possible to precisely target the desired growth stage. The general character of branches is controlled using a small number of parameters. The generated branches can be pruned, bent and stretched to meet the requirements of the design (Figure 2). Branch widths are defined procedurally, but can be modified by the user by placing constraints at any point throughout the tree. Saved trees can be reloaded and further manipulated, thus providing a set of templates for different tree types that expands as the system is used. By combining fast procedural methods with an intuitive interface, TreeSketch makes it possible to quickly create diverse natural and artistically expressive trees.

Following a review of related work (Section 2), we present the interface design and algorithms underlying the tree generation and editing techniques implemented in TreeSketch (Section 3). We then present select examples of the application of the system (Section 4), and conclude the paper with a discussion of the results and suggestions for further work (Section 5).

2. Background

Algorithms for procedural tree modeling form a continuum spanning two extremes: those generating trees as recursive or hierarchical branching patterns, and those in which branching patterns emerge from the competition of candidate limbs for space or light. The middle ground is occupied by algorithms that combine both aspects in various proportions. We review the spectrum of procedural models by focusing on their capability to generate trees with a realistic appearance and capacity for interactive control of tree form.

2.1. Recursive and hierarchical models

Generative algorithms. Procedural trees are often generated using recursive or hierarchical algorithms. Most older algorithms [Hon71, AK84, Blo85, Opp86, PL90] operate in a bottom-up fashion, *i.e.*, by repeating a locally defined ramified geometry over consecutive orders of the branching hierarchy. Recursive and hierarchical algorithms can yield a wide range of tree forms [dEF*88], but the modeler's ability to control them is impeded by the need to understand the intricacies of the generative processes and their software implementation. These difficulties have been partially circumvented by limiting the modeler to manipulating a set of exposed parameters using standard control panels [Opp86, PL90] or specialized graphical editors [IOOI05, IOI06b]. Additional techniques include confining modifications of the algorithm to the choice between predefined options [dEF*88], introducing dedicated modeling languages to specify the generative algorithms [PL90, KP03], and introducing graphical interfaces for composing the algorithm from predefined components [LD99]. In most cases, however, global attributes of the generated structures — the overall silhouette of the tree, the shape of key limbs, and the density of branches — emerge from the execution of the bottom-up algorithms, and are not directly controlled. One exception is the method for modeling topiary trees [PJM94], in which branches that grow outside a predefined shape are repetitively pruned. The resulting trees, however, have the artificial appearance of topiary trees, rather than that of natural trees that happened to grow into particular forms. Another exception is the method for inferring structures matching a high-level specification (e.g., a sketch) from a range of possibilities afforded by the underlying generative algorithm (a grammar) [TLL*11].

Control of silhouette. The above shortcomings were addressed in top-down algorithms, which operate by decomposing plant axes into smaller segments (internodes), as opposed to composing them from the internodes [PMKL01]. The key observation was that the silhouette of a tree with a well defined trunk is largely determined by the reach of its

first-order branches, which therefore can be inferred from the silhouette of the tree [RB85]. Methods proposed to define silhouettes include specialized surfaces controlled by a small number of parameters [RB85, WP95, BPF*03] and arbitrary surfaces of revolution with a graphically defined profile [DL97, PMKL01]. Boudon et al. [BPF*03] extended the top-down approach by introducing a hierarchy of envelopes to define both the silhouette of the whole tree and the silhouettes of individual branches. More recently, Wither et al. [WBCG09] used a sketch-based interface to define 2D silhouettes of branches that were subsequently rearranged in 3D.

Control of limb shape. The top-down systems by Deussen and Lintermann [DL97], Prusinkiewicz et al. [PMKL01] and Boudon et al. [BPF*03] allowed the modeler to specify the shape of selected axes using a parametric curve editor. A more intuitive method was introduced by Okabe et al. [OOI05] and Ijiri et al. [IOI06b], who defined the axes of two-dimensional branching structures by sketching. Diverse methods were proposed to infer the three-dimensional shape of plant axes from 2D strokes. They employed the assumption of constant stem curvature in 3D [IOOI05], projections of sketches onto billboards arranged in 3D, and multiple projections [IOI06a]. Onishi et al. [OMKK06] bypassed the problem of inferring 3D axes from 2D sketches by using a 3D input stylus. Complementing these techniques, Power et al. [PBBPS99] introduced interactive pruning and bending of branches using inverse kinematics.

Control of branch distribution and proliferation. The problem of inferring 3D form from 2D sketches extends from the control of limb shape to the layout of branches. The 3D arrangement of branches has been inferred from 2D sketches by rearranging branches to maximize their mutual distances [OOI05] or imposing a phyllotactic pattern of branch arrangement [WBCG09]. In the technique introduced by Chen et al. [CNX*08], a sketch representing main branches and, optionally, the desired silhouette of the tree is compared with a database of reference trees, and the best matching tree provides parameters for a recursive generative algorithm. Related techniques have been proposed to distribute and proliferate small branches while reconstructing trees from photographs or laser scan data. For example, Tan et al. [TZW*07] inferred parameters used for branch proliferation from the layout of visible branches. Livny et al. [LPC*11] proliferated branches by rearranging templates generated with L-systems to fill given envelopes.

2.2. Self-organizing trees

Generative algorithms. The main limitation of recursive and hierarchical models is that the harmonious distribution of branch densities throughout the tree crown is not achieved automatically and must be monitored carefully by the modeler. This problem is much reduced in self-organizing models, which emphasize competition between branches as a fundamental process controlling branch proliferation in nature (as proposed from a biological perspective by Ward [War09], Sachs and Novoplansky [SN95], and Sachs [Sac04]).

Computational models based on the self-organizing principle have their roots in early models of branching structures proposed by Ulam [Ula62] and Cohen [Coh67], and within computer graphics in methods that captured the role of light [Gre89, COMM94] and space [RCSL03, RLP07] in shaping the tree. In the tree models proposed by Měch and Prusinkiewicz [MP96], control of branch density is not limited to local action, but is integrated globally through internal signaling. The signals integrate the amount of light reaching entire branches to decide whether they were profitable to the plant and should be kept, or whether they were a liability and should be shed. More recently, Pałubicki et al. [PHL*09] extended this idea with a model in which the internal flow of signals also biases the outcome of competition between buds for light. Parameters controlling competitive bias and shedding capture tree characteristics that are recognized as visually important by arborists, such as the presence or absence of a well-defined trunk.

Control of tree form. Explicit representation of space in self-organizing models makes it possible to control the form of plants by manipulating their environment. An early example was given by Měch and Prusinkiewicz [MP96], who used a paint program to define the distribution of water in the soil that guided the development of a branching root structure. Rodkaew [RCSL03] and Runions et al. [RLP07] observed that in self-organizing trees, tree silhouettes can easily be specified by constraining the space within which the trees grow. Neubert et al. [NFD07] extended Rodkaew's algorithm to incorporate branches that approximate a given input obtained from photographs. Likewise, Côté et al. [CWFV09] adapted the space colonization algorithm of Runions et al. to proliferate branches around main limbs obtained from LIDAR data. Beneš et al. [BvMM11] divided the environment into a set of regions guiding the form of individual, possibly distinct components of the tree, such as naked branches and leaf clusters.

In the context of interactive systems, the environment may be regarded as a two- or three-dimensional canvas, in which the user paints properties with a brush. For example, Maya PaintEffects provides a brush that makes it possible to indicate the region and direction of growth of a population of simple flowers. Zakaria and Shkuri [ZS07], and Pałubicki et al. [PHL*09] proposed a procedural brush to indicate regions in which the tree grows. Due to the fast response of the underlying tree-generating algorithms, the modeler has the impression of interactively guiding tree growth.

3. Design and implementation of TreeSketch

The procedural component of TreeSketch is based on the algorithms for generating self-organizing trees described by

Figure 3: *Main components of the TreeSketch interface. Left: the workspace with a modeled tree. Right: The view with a control panel pulled down.*

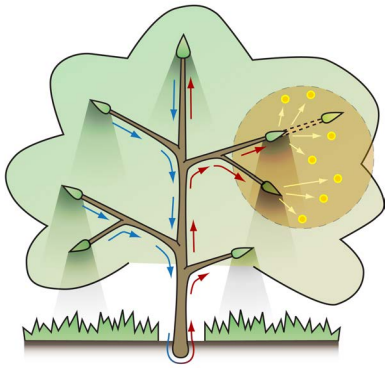

Figure 4: *A conceptual model of tree development implemented in TreeSketch. Higher-positioned branches cast shadows on lower ones (here shadows are shown only for buds). Information about the light exposure of each bud, combined with information on the location of the buds on their supporting axes, propagates towards the tree base (blue arrows) and guides the distribution of a growth-inducing vigor signal flowing back to the buds (red arrows). A bud that has sufficient vigor and wins the competition for some markers of free space (yellow dots pointed to by arrows) produces a new shoot (dashed lines). The markers are distributed with a brush (brown circle) or a lasso.*

Runions et al. [RLP07] and Pałubicki et al. [PHL*09]. These algorithms have been tailored for interactive use, taking full advantage of a multi-touch tablet interface, and implemented on an Apple iPad 2.

3.1. A modeler's perspective

Almost the entire screen is devoted to the *workspace* (Figure 3, left), in which the modeler can view, grow, sketch, brush, and edit the tree using a variety of one- and two-handed gestures. Sliding two fingers across the screen translates the tree, the standard two-touch pinch gesture zooms it in and out, and moving three fingers across the screen rotates it in 3D. Single-stroke gestures operate the brush that guides the growth of the tree. The *rotation strip* at the bottom of the screen makes it possible to rotate the tree with one hand while it is being modeled with the other hand. More specialized gestures will be presented in the context of the operations they perform.

The *side bar* collects buttons that are frequently activated while the modeler sketches or manipulates the tree. Pressing the Brush button opens a slider that controls the diameter of the brush. The Grow button opens another slider, which controls the progress of simulation time while the tree grows autonomously, and makes it possible to reverse the progress of time to ungrow the tree. Lasso changes the interpretation of single-finger strokes, making it possible to sketch silhouettes of the whole tree or its parts. The Width button is used to place and manipulate widgets for controlling the diameter of selected branches. Multi-level Undo cancels previous actions. The side bar can be switched between the left and right side of the screen to accommodate operations with either hand.

The *tool bar* above the workspace is used to select and pull down any of the five *control panels*. With a panel pulled down (Figure 3, right), the workspace is reduced in size, but remains active. Two panels, Architecture and Brush, contain manipulable diagrams and sliders for controlling procedural aspects of the model. The remaining panels, Leaves, Branches and Background, control details of the form and the rendering of the tree on the tablet. The tool bar also includes buttons for hiding all elements of the interface while previewing the tree, saving designed trees, accessing the repository of saved trees, initiating a new design, and displaying help pages.

3.2. Tree generation

Overview of the algorithm. Pałubicki et al. [PHL*09] presented two classes of self-organizing models. Models employing competition for light were shown to generate plausible forms of natural trees. By incorporating an internal signaling mechanism that biased the results of competition, these forms could be controlled on the important scale from excurrent to decurrent forms (i.e., with and without a conspicuous trunk). In contrast, competition for space produced a narrower range of plausible trees, but allowed for interactive guidance of tree development with a brush. In TreeSketch, we aimed at combining the potential for high realism with interactive control of tree form. To this end, we created a model in which development depends *simultaneously* on light and space. The resulting conceptual model for TreeSketch is shown in Figure 4. Beginning with a seedling, branches develop from buds distributed along their supporting axes. Whether a bud will develop into a shoot, and how

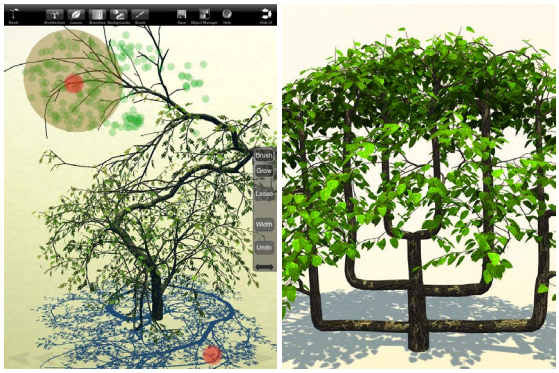

Figure 5: *Brushing and sketching. Left: A snapshot of the TreeSketch screen. The tree was rotated while brushing, resulting in a helical trunk. Pink dots indicate position of fingers on the tablet. Right: A candelabra espalier created by sketching the main branches, then brushing the upper crown.*

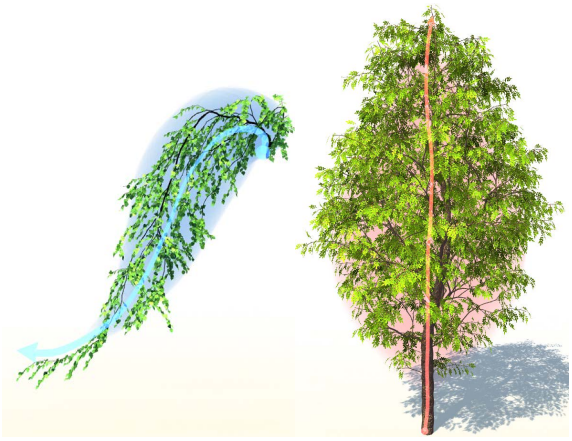

Figure 6: *A hanging branch and a tree created with single strokes of the brush (arrow) with interactively changed radius (visualized as colored sweeps).*

long this shoot will be, depends on the vigor of the bud and the availability of space. Model parameters may bias vigor distribution towards buds exposed to a higher amount of light vs. those in less light, terminal vs. lateral buds, and lateral buds positioned on the upper side of the mother branches vs. those located underneath, yielding different tree architectures.

Brushing and sketching. The space available for growth is represented as a set of marker points (yellow dots in Figure 4) [RFL*05, RLP07, PHL*09]. The modeler can guide growth interactively by distributing these points using a procedural brush or by defining or extending the envelope of the tree with a lasso tool. Marker points can also be generated automatically in the general vicinity of the buds, to simulate natural growth.

The *brush* is implemented by continually generating marker points (with uniform random distribution) within a sphere invoked and manipulated by the modeler using single-finger strokes (Figure 5, left). By default, the brush moves in the plane that passes through the base of the tree and is parallel to the screen. Strokes originating at a branch modify the depth of the drawing plane so that it includes the selected branch point. The radius of the brush is controlled by a slider activated with the Brush button on the side bar, and can be changed while brushing (Figure 6). With the radius of the brush decreasing to zero, brushing transitions in a continuous manner to sketching (Figure 5, right).

Similar rules apply when the *lasso* is used to sketch the silhouette of the entire tree or its part (Figure 7). To infer a 3D envelope from a 2D silhouette sketch, image-based erosion of the silhouette is first used to infer its chordal axes [Hal89]. The silhouette is then inflated around these axes as in the Teddy system [IMT99]. The inflated envelope is filled with marker points by generating random points within a cube bounding the envelope, and retaining those points that fall within the envelope.

Competition for space. The mapping of a distribution of markers into a tree form was described before [RLP07, PHL*09], but its refinement in TreeSketch includes many details that are important to the "feel" of the interface, and consistency between trees modeled interactively and growing autonomously. Consequently, we describe it here in detail.

Branches grow in the directions that are initially determined by the distribution of buds on their supporting axes (Figure 8A), and modified by gravity and the distribution of available space. Each bud is surrounded by a sphere of radius r, representing the zone occupied by the bud. Extending beyond this zone is the bud's volume of perception: a truncated cone (with a spherical base) characterized by the angle of perception ζ and radius of perception $r+d$ (Figure 8B, see also [PHL*09]). At the beginning of each simulation step, the set M of markers of free space is augmented with new markers generated with the brush, the lasso tool, or autonomously. In the autonomous case, the markers are generated within the perception volume of each bud. Next, the following markers are removed from the set M:

- Markers that are within the occupancy zone of one or more buds (e.g., Marker 1 in Figure 8B).
- Markers that were generated with a brush and are older than the maximum age. The removal of lingering markers increases the modeler's sense of direct, immediate guidance of growth.
- Markers generated with a lasso or autonomously, which persist after the tree has stopped growing. Such a situation may occur when no markers are within the volume of perception of any bud.

Figure 7: *Examples of tree forms controlled with a lasso (red contours shown for two of the models).*

If any markers remain, the buds compete for them according to the following rules:

- A marker within the perception volume of a single bud is associated with this bud (Markers 2 and 3 in Fig. 8B);
- A marker within the intersecting volume of two or more buds is associated with the closest of these buds (Marker 4);
- The markers outside of the volume of perception of any bud remain unassociated in a given simulation step (Marker 5).

By default, all buds associated with at least one marker are considered *enabled*: they will produce new shoots if they are

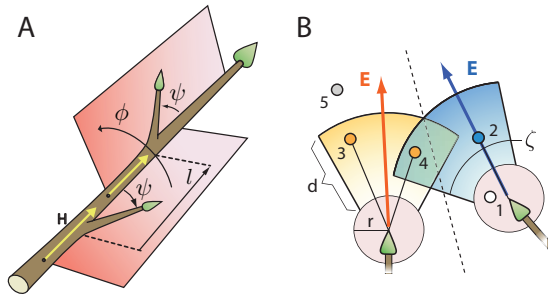

Figure 8: *Buds in space. A) Distribution of buds and lateral branches is defined by the internode length l, phyllotactic angle ϕ and branching angle ψ. At the branching point, the main branch may deflect in the direction opposite to the lateral bud by angle δ (not shown). B) Competition of buds for space. Circles of radius r represent the spherical zones occupied by buds. The colored areas represent each bud's volume of perception, characterized by radius $r+d$ and angle ζ. Dots represent markers of free space, for which buds compete (details in the text). The average direction towards the markers associated with a bud defines its preferred growth direction* $\boldsymbol{E}$.

sufficiently vigorous. The resulting proliferation of shoots from the existing branches can make it difficult, however, to create a large new branch in proximity of one or more older branches. Consequently, TreeSketch also supports the *selective growth mode*, in which growth is limited to the branch originating at the bud selected by the modeler. To enter this mode, the modeler presses and holds a bud until the selection is confirmed by expanding red circles. In the selective mode, all buds remove markers within their occupancy zones, but only buds in the new branch compete for the remaining markers and may grow (Figure 9).

Acceleration of proximity queries. The search for buds associated with marker points is accelerated in TreeSketch by placing the tree in a voxel space. The voxel size (edge length) is equal to the bud's radius of perception, $r+d$. Each voxel stores the list of buds it contains. These lists are updated incrementally when the buds are created, removed, or change position due to an interactive manipulation of branches. Since a marker may only influence a bud within the radius $r+d$, it suffices to consider the 27 voxels surrounding each marker to determine the bud it may affect.

Shoot growth. Superimposed on the competition for space is a process evaluating the vigor of buds, which determines which of the enabled buds will actually produce new shoots, and how long these shoots will be. We model this process using the extended Borchert-Honda (BH) algorithm described by Pałubicki et al. [PHL*09] (see Figure 4 for intuition), with the following modifications.

- Instead of using the amount of light L reaching buds directly as the input Q to the BH algorithm, we use the relation $Q = bL^{\kappa}$. Parameter $\kappa > 0$, set by the modeler, controls the sensitivity of buds to light. Allowing only buds exposed to strong light to produce new branches reduces the subsequent shedding of branches. This reduction is desirable during interactive modeling, as shedding can inter-

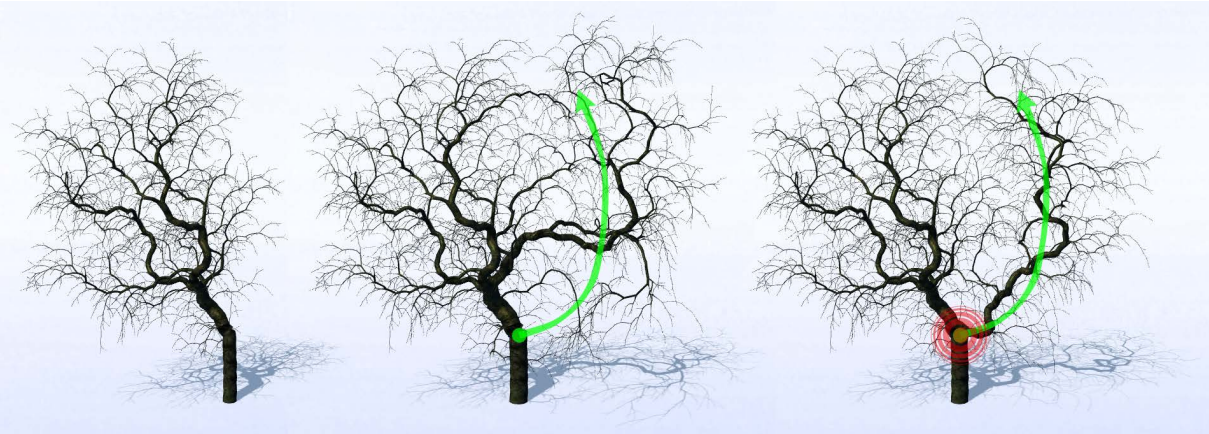

Figure 9: *Comparison of non-selective and selective growth. The initial tree crown (left) was refined by non-selective growth (middle) and selective growth (right). Non-selective growth produced a series of branches extending from the initial tree lobe, whereas selective growth produced a separate lobe from the selected bud indicated by the expanding red circles.*

fere with the intentions of the modeler. The role of variable b is discussed in the section on gravimorphism below.

- Instead of using the amounts of resource v output by the BH algorithm to determine the number of metamers n directly with the formula $n = \lfloor v \rfloor$, we use a scaled version of this formula, $n = \lfloor \frac{v}{v_{max}} n_{max} \rfloor$. Here v_{max} is the highest vigor of any enabled bud and n_{max} is a parameter, set by the modeler, which determines the maximum shoot length. This change ensures that the brush remains effective in lower parts of the tree, which receive little light. Increased values of parameter n_{max} result in wispy branches, such as those often found in pendulous trees (e.g., Figure 1, center).

Shedding. Shedding of branches is an important aspect of tree development, controlling the density of the crown and the self-pruning of branches in lower parts of the trunk. In TreeSketch, a branch is shed if the ratio of its vigor to the size of the branch (measured in the number of internodes) falls below a user-defined threshold Th.

Gravimorphism. The coefficient of proportionality b in Equation $Q = bL^{\kappa}$ captures the differential development of lateral buds depending on their orientation on horizontal or slanted parent axes — an aspect of gravimorphism. In the case of *epitony*, buds on the upper side of such axes are favored, *amphitony* favors buds in the most horizontal position, and *hypotony* favors buds on the lower side of the parent axes [BC07]. To quantify these phenomena, we define an ellipse lying in a plane perpendicular to the parent axis at the location of a bud (Figure 10A). The propensity b of this bud to grow depends on the distance d between the branch axis and the ellipse, measured in the direction χ determined by the polar position of the bud on the axis (Figure 10B). An additional factor is the angle τ between the axis direction $\boldsymbol{H}$ and the vertical direction $\boldsymbol{Y}$. Grouping these factors together,

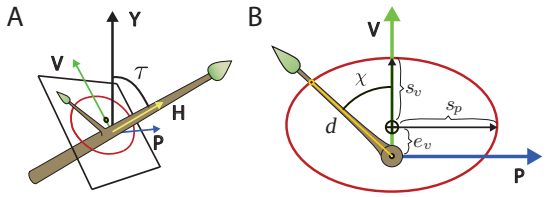

Figure 10: *Specification of the gravimorphic response. The axes s_v and s_p of the control ellipse are aligned with the axes $\boldsymbol{V}$ and $\boldsymbol{P}$ of the $\boldsymbol{HVP}$ ($\boldsymbol{H}$eading – most $\boldsymbol{V}$ertical – $\boldsymbol{P}$arallel to the ground) reference system associated with the internode. The center of the ellipse is translated by distance e_v along the axis $\boldsymbol{V}$. Bias of bud activation in direction χ is proportional to the distance d from the internode to the ellipse, measured in this direction.*

the effect of gravimorphism is captured using the equation

$$b = \cos^2 \tau + d \sin^2 \tau, \qquad (1)$$

which maximizes this effect for horizontal parent axes and reduces it to zero for vertical axes. Gravimorphic responses of different types and magnitudes can be defined by manipulating the position and displacement of the control ellipse (Section 3.4). Their impact on tree forms is best illustrated in connection with gravitropism.

Gravitropism. The term *gravitropism* refers to the tendency of branches to maintain a preferred orientation with respect to gravity. Traditionally, this tendency was characterized qualitatively, using terms such as negative gravitropism (the tendency of branches to grow upwards), plagiotropism (the tendency to maintain a slanted or horizontal orientation) and positive gravitropism (the tendency to grow downward). Digby and Firn [DF95] quantified and unified these notions

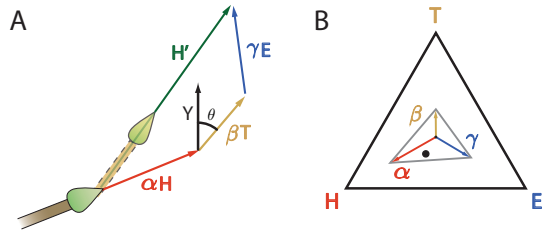

Figure 11: *A) Calculation of tropic responses. New growth direction $\boldsymbol{H'}$ is a weighted sum of the current bud direction $\boldsymbol{H}$, the tropic vector $\boldsymbol{T}$, and the preferred growth direction $\boldsymbol{E}$ returned by the environment. B) Visualization of weights α, β, γ as a point (black dot) in a triangle with vertices labelled $\boldsymbol{HTE}$ and as vectors from the gravity center of this triangle to its vertices.*

Figure 12: *Contrasting tree forms resulting from the interplay between gravitropism and gravimorphism. Left: downward tropism + epitony; right: upward tropism + hypotony.*

by introducing the *gravitropic set-point angle* (GSA): the angle θ to the vertical direction $\boldsymbol{Y}$ that the branch axes tend to maintain. To model tropisms in TreeSketch, we orient consecutive internodes of a growing axis so as to approach a user-specified GSA. First, we calculate the local tropism vector $\boldsymbol{T}$ by rotating the global up vector $\boldsymbol{Y}$ by θ in the vertical plane including the current growth direction $\boldsymbol{H}$. If $\boldsymbol{H}$ is almost vertical, we break symmetry by choosing the vertical plane of rotation at random. As shown in Figure 11A, the new growth direction, $\boldsymbol{H'}$ is then calculated as the weighted sum

$$\boldsymbol{H'} = \alpha\boldsymbol{H} + \beta\boldsymbol{T} + \gamma\boldsymbol{E}, \qquad \alpha+\beta+\gamma = 1, \tag{2}$$

where $\boldsymbol{E}$ is the unit vector corresponding to the optimal direction of local growth obtained from the environment (Figure 8B). Parameters α, β and γ control the relative influence of current axis direction, gravitropism and space available for growth, respectively. Their values can be conveniently visualized as a point in barycentric coordinates spanned by vertices $(\alpha,\beta,\gamma) = (1,0,0)$, (0,1,0) and (0,0,1), respectively (Figure 11B). This visualization is used in the design of the widget for manipulating parameters α, β and γ interactively (Section 3.4). Gravimorphism and gravitropism often work in concert, producing dramatically different tree forms. Examples are shown in Figure 12.

3.3. Tree manipulation

Interaction with procedural tree models can be accomplished not only by guiding tree development, but also by editing trees that have already been generated. TreeSketch supports four editing operations: modifications of branch shape, selective modification of branch width, pruning, and undo.

Modification of branch shape. Branch bending provides a convenient metaphor for editing trees, consistent with the physical manipulations employed by horticulturalists when training trees. In computer graphics, bending was introduced as a method for editing plants by Power et al. [PBBPS99], who implemented it using inverse kinematics. We reimplemented and tested that technique in an early version of TreeSketch. It produced plausible deformations, but suffered from two limitations. First, it was not reversible: moving the end effector back to its original position through a circular path did not restore the branch to its original form. Second, manipulations did not preserve the local character of branches.

In the broader context of surface editing, similar limitations were overcome in the PriMo deformation method [BPGK06]. In TreeSketch, we have adapted PriMo to tree skeletons. PriMo represents mesh geometry as a set of rigid prisms connected by elastic joints (Figure 13A). The modeler manipulates this structure by placing positional constraints on a subset of these prisms. The resulting deformation is determined by minimizing the elastic energy of the joints. In our system, prisms correspond to the internodes in the path between a modeler-selected base internode B and end internode E. The end internode can be freely chosen within the subtree rooted at B, and further internodes can be selected within the path from B to E. The tree is manipulated by changing the position of the selected internodes with a multi-touch gesture (Figure 14). As an axis is deformed, other branches maintain their relative positions and orientations with respect to their supporting internodes.

A tunable parameter of the PriMo model allows for separately controlling the susceptibility of joints to stretching and bending [BPGK06]. Although real tree branches are almost non-stretchable, we found it convenient to allow for stretching when editing branches in TreeSketch.

Branch width specification. By default, branch width in TreeSketch is determined procedurally, using the da Vinci formula $d^n = d_1^n + d_2^n$ that relates the diameter d of an internode below a branching point to the diameters d_1 and d_2 of the internodes above [Mac83]. This formula makes it possible to recursively compute the width of all branches by propagating information from the extremities of the tree towards the trunk. The trunk of a tree supporting n terminal

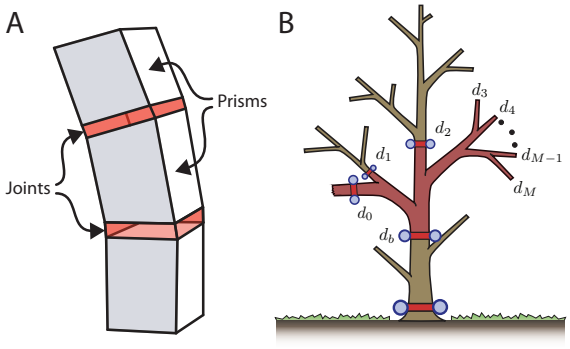

Figure 13: *A) Representation of branch geometry for the PriMo deformation method. Prisms correspond to internodes, joints correspond to nodes of a path in a tree. B) Width constraints (red) divide a tree into subtrees. Within each subtree, Equation 3 is solved to yield the exponent n, then branch width is calculated using the da Vinci formula.*

Figure 14: *Bending a branch with a three-touch gesture. Left: the initial state of the system; right: the result of manipulation. The bottom and top touches establish the base B and end effector E, respectively. The in-between touch provides an additional positional constraint.*

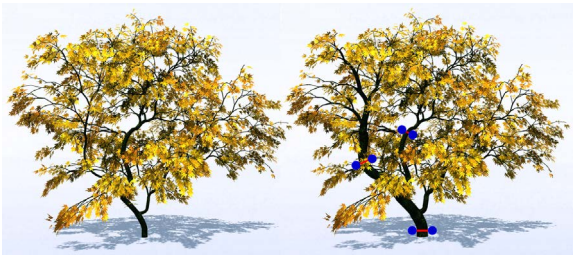

Figure 15: *A model (left) is refined by modifying the width of selected branches (right).*

branches with diameters $d_0, \ldots, d_M$ will thus have diameter d that satisfies the equation

$$d^n = \sum_{i=0}^{M} d_i^n. \tag{3}$$

Da Vinci claimed that the cross-section of the parent branch is equal to the sum of cross-sections of the child branches, which implies that the exponent n involved in this equation is equal to 2. MacDonald [Mac83] pointed out, however, that other values of this exponent, usually between 2 and 3, may be more realistic. In TreeSketch, we use this degree of freedom to incorporate user-defined constraints into the computation of branch width. In the simplest case, the user specifies the width of branches at the extremities (d_e) and at the base of the tree (d_b). From Equation 3 it then follows that the exponent n is equal to $\frac{\log M}{\log d_b - \log d_e}$. In general, the modeler may place an arbitrary number of constraints at arbitrary locations on the tree (Figure 13). These locations partition the tree into subtrees and Equation 3 is applied to calculate the exponent n separately in each subtree. As the branches at the extremities of these subtrees may have different diameters d_e, Equation 3 no longer has an analytic solution and is solved numerically, using the Newton-Raphson method. The computation is fast, allowing branch widths to be specified interactively even for complex trees with a large number of constraints. Selective changes in branch width have a significant impact on the appearance of the tree (Figures 2 and 15).

Pruning. Pruning is the most common tree manipulation procedure in horticultural practice. In TreeSketch it is accomplished by double-tapping on a branch. The branch is then removed at the branching point closest to the tapped location, and the diameter of remaining branches is adjusted as if the pruned branch was never present. When needed, the stub of the removed branch can be maintained, and the radius of the remaining branches preserved, by placing a width constraint on the branch, and pruning just above it (e.g., the stub over constraint d_0 in Figure 13).

Undo. To support undo operations, all internodes of the tree are time-stamped by the iteration number of the generative process in which they have been created. Branches that have been shed by the tree or pruned are not discarded, but kept on a stack. Furthermore, all editing events are time-stamped and also saved on a stack. In the case of branch manipulation, the saved information includes the positions of all internodes in the path from the base B to the end effector E before the branch has been modified. Continuous undo is implemented by rolling back the state of the tree to an earlier time by small intervals. Discrete undo rolls back the model by entire operations, e.g., a stroke of a brush. The undo operations are controlled by a slider activated with the Grow button and the Undo button on the side bar, respectively.

3.4. Interface design

The general appearance of the interface has been outlined in Section 3.1 and illustrated in Figure 3. The gestures for operating the brush, the lasso, and editing the tree, have been explained in the context of these operations. The key remaining aspect of the interface is the architectural panel (Figure 16), which allows the modeler to control general characteristics of the modeled trees by manipulating key parameters of the generative algorithm. Visual feedback is provided by a schematic two-dimensional tree, generated by the same procedural model as the "main" tree being created, which changes form instantaneously according to the parameters used. These parameters are manipulated with five special-purpose widgets, which control:

1. Tropism direction. This widget is represented by an arrow superimposed on the schematic tree model and controls the gravitropic set-point angle θ shown in Figure 11A .
2. Gravimorphism. This widget controls the preferential growth of buds according to their orientation on the supported axes. It is a manipulable version of the ellipse in Figure 10B, returning values of parameters s_v, s_p and e_v.
3. Branching angle ψ and deflection angle δ (Figure 8A).
4. Branch direction. This widget controls the relative impact of the current direction, tropism and markers of empty space on the direction shoot growth. It is a manipulable version of Figure 11B, returning values of weights α, β and γ.
5. The interaction of buds with the markers of free space. This widget is a manipulable version of Figure 8B, returning values of parameters r, d and ζ.

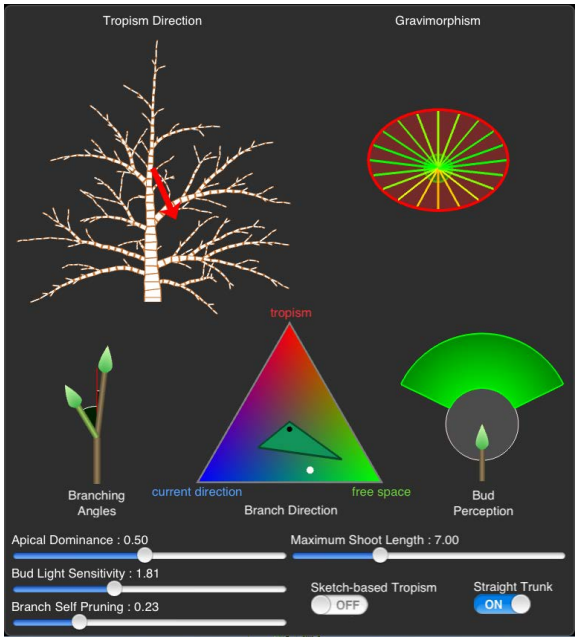

Figure 16: *The architectural panel.*

In addition, apical dominance λ [PHL*09], sensitivity of buds to light κ, branch shedding threshold Th and maximum shoot length n_{max} are controlled using sliders. Finally, the panel includes switches for inferring tropism direction from the direction of brush stroke (particularly useful when modeling pendulous tree forms) and forcing upward tropism of the trunk independently of the tropism of the branches (useful when modeling conifers). The ranges and default values of key parameters are provided in the supplementary table.

3.5. Model visualization

Trees modeled with TreeSketch are intended to be assembled into scenes and rendered using external programs. Nevertheless, realistic rendering is an important component of the interactive modeling process as well. To this end, we combine the following rendering techniques:

- Phong light model of the tree, with texture-mapped leaves and texture- and normal-mapped bark. Position of the light source is indicated by a spherical "sun", which can be interactively moved around the tree. By default, rotation of the tree also rotates the sun. Alternatively, the modeler may pin the position of the sun with one hand while rotating the tree with the other.
- Indirect light model. An approximate distribution of light intensity within the tree crown is computed as an inherent component of the growth model. We use this information while rendering, by modulating the ambient light component in the Phong model. This results in a much improved perception of crown form in three dimensions at no computational cost.
- Shadows. Shadow-mapping is used to cast Gaussian-filtered blurred shadows on the ground and hard shadows within the tree.

In addition, we account for differences in leaf color by shifting the hue of leaves on less vigorous branches towards yellows and reds. The magnitude of this shift is a parameter controlled by the modeler.

Speed considerations. To accelerate rendering, several quality-improving features, such as anti-aliasing and soft shadows, are temporarily disabled while interactively manipulating the tree. These effects are automatically phased in when the modeler is not interacting with the system.

4. Examples

Diverse trees modeled with TreeSketch are shown in Figures 17. While specific aspects of their form are due to the use of the brush and lasso, general architectural characteristics correspond to the parameters grouped in the architectural panel. The impact of tropism is most easily discernible, ranging from the upward tropism (trees B3 and B4) to plagiotropism (A1, A4, B1) to strong downward tropism (B2, C2). In the case of trees A3 and C1, the initial growth with

Figure 17: *Diverse trees modeled using TreeSketch.*

upward tropism was followed by plagiotropic growth. Trees B2 and B3 illustrate the effect of strong gravimorphism, favoring growth on the upper and lower side of the parent branches, respectively. The sparsity of tree C1 is due to the strong shedding of branches, while the sparse, gnarled appearance of tree C4 is due to a combination of high sensitivity to light and interactively increased branch width.

Figure 18 illustrates the complexity and visual realism of structures generated with TreeSketch. The tree on the left also illustrates the effectiveness of the indirect light model, which has been used almost exclusively in rendering this tree.

Figure 19 provides examples of generating realistic tree models that address the artistic requirements of target scenes. The tree on the left is a weathered tree on a sheer cliff face. The tree's architecture was modeled by combining direct sketching of the main branches with procedural brushing of the tree crown. The form of branches was then modified to generate a more gnarled appearance using bending. Finally, width constraints were used to increase the bulk of the trunk and the leafless branches near the base, and create a more even distribution of widths in the crown. A similar design process was used to create the gnarled oak tree in Figure 19 on the right.

5. Conclusions

In spite of decades of computer graphics research, the modeling of three-dimensional objects remains a tedious task. The crux of the problem is the tension between detailed control of form by the modeler and the complexity of the objects being modeled. We have presented a program for modeling trees that reconciles the interactive control needed in creative design with the emergence of form inherent in procedural generation. This synthesis was accomplished by integrating

Figure 18: *Examples of complex tree structures modeled and rendered with TreeSketch.*

appropriately tailored procedural methods with an interface taking ample advantage of multi-touch displays. Our work demonstrates that:

- procedural (tree) models can easily be controlled by modelers without a computer science background,
- tree modeling based on a sound biological basis does not require a biological background on the part of the modelers, and
- complex tree models can be created procedurally at interactive rates using current (relatively low-end) hardware.

These insights are not obvious and contradict statements routinely made in the literature.

We are continually astonished by the degree and intuitive feel of control afforded by strokes of different length, direction, speed, and brush width. While rigorous user studies of a program for performing tasks as complex as artistic design of trees are difficult, our perception of TreeSketch is supported by positive feedback from the users of two earlier versions of TreeSketch (97 comments on 13,000 downloads from iTunes between March and December, 2011). They found that TreeSketch presents “excellent balance between procedural and hand-crafted controls", “combines intuitive modeling of plant life [...] with a simplicity and joy of a game", and is “easy to learn and use, [yet] challenging to master" because of the depth of artistic possibilities it offers.

The applications that we originally envisioned for TreeSketch lie in the domain of image synthesis for computer animation, games, and computer-assisted landscape design. Experimenting with TreeSketch we were surprised, however, by the extent to which it has sharpened our own perception of tree form and development in nature. We thus also intend to explore possible applications of TreeSketch to teach students of botany and art, including children, about tree development and form. Another prospective application of TreeSketch is in the domain of horticulture. In this application, the user would sketch the form of an existing tree using model parameters consistent with the tree species, edit the form by pruning, then grow it autonomously to predict the impact of pruning on the tree over the years.

A number of problems remain open for further research. Botanists observed that not only the inclination of branches, but also their curvature play a role in gravimorphism. The incorporation of this role of curvature may further increase the diversity and verisimilitude of trees modeled with TreeSketch. Furthermore, flowers and fruits are an essen-

Figure 19: *Two scenes with trees generated using TreeSketch. The scenes were assembled and rendered in Maya.*

tial component of a tree's appearance. Architectural analysis and mathematical models make it possible to predict the positions of these organs, providing a basis for their incorporation in the models. The distribution and orientation of leaves also deserve a more careful consideration. Branches are currently modeled as generalized cylinders, and branching geometry simply results from their intersections. More refined methods have been described, and possibly could be incorporated into TreeSketch without sacrificing the speed of model generation. Finally, the TreeSketch interface is an early exploration of possibilities opened by multi-touch tablets in the domain of interactive procedural modeling. We expect many further advances in this area.

6. Acknowledgements

We thank Thomas Burt for the tool bar icons, Helen Ai He for the swan model in Figure 7, Wojtek Pałubicki and the participants of the 2011 Bellairs Workshop on Computer Animation for insightful comments on a prototype of TreeSketch, Christophe Godin for discussions and for hosting SL during his visit to Montpellier, Lynn Mercer and Brendan Lane for editorial input, and the anonymous reviewers for helpful comments. We also gratefully acknowledge the support of this research by the Graphics, Animation and New Media Network of Centers of Excellence, Natural Sciences and Engineering Research Council of Canada, and INRIA international program Equipes Associées.

References

[AK84] AONO M., KUNII T. L.: Botanical tree image generation. *IEEE Computer Graphics and Applications 4*, 5 (1984), 10–34. 2

[BC07] BARTHÉLÉMY D., CARAGLIO Y.: Plant architecture: A dynamic, multilevel and comprehensive approach to plant form, structure, and ontology. *Annals of Botany 99* (2007), 375–407. 7

[Blo85] BLOOMENTHAL J.: Modeling the Mighty Maple. *Computer Graphics 19*, 3 (1985), 305–311. 2

[BPF*03] BOUDON F., PRUSINKIEWICZ P., FEDERL P., GODIN C., KARWOWSKI R.: Interactive design of bonsai tree models. *Computer Graphics Forum 22*, 3 (2003), 591–599. 3

[BPGK06] BOTSCH M., PAULY M., GROSS M., KOBBELT L.: Primo: Coupled prisms for intuitive surface modeling. In *Proceedings of the fourth Eurographics symposium on Geometry processing* (2006), pp. 11–20. 8

[BvMM11] BENEŠ B., ŠT'AVA O., MĚCH R., MILLER G.: Guided procedural modeling. *Computer Graphics Forum 30*, 2 (2011), 325–334. 3

[CNX*08] CHEN X., NEUBERT B., XU Y.-Q., DEUSSEN O., KANG S. B.: Sketch-based tree modeling using Markov random field. *ACM Transactions on Graphics 27*, 5 (2008), 109. 3

[Coh67] COHEN D.: Computer simulation of biological pattern generation processes. *Nature 216* (1967), 246–248. 3

[COMM94] CHIBA N., OHKAWA S., MURAOKA K., MIURA M.: Visual simulation of botanical trees based on virtual heliotropism and dormancy break. *The Journal of Visualization and Computer Animation 5* (1994), 3–15. 3

[CWFV09] CÔTÉ J.-F., WIDLOWSKI J.-L., FOURNIER R., VERSTRAETE M.: The structural and radiative consistency of three-dimensional tree reconstructions from terrestrial LIDAR. *Remote Sensing of Environment 113* (2009), 1067–1081. 3

[dEF*88] DE REFFYE P., EDELIN C., FRANÇON J., JAEGER M., PUECH C.: Plant models faithful to botanical structure and development. *Computer Graphics 22*, 4 (1988), 151–158. 2

[DF95] DIGBY J., FIRN R. D.: The gravitropic set-point angle (GSA): The identification of an important developmentally controlled variable governing plant architecture. *Plant, Cell & Environment 18* (1995), 1434–1440. 7

[DL97] DEUSSEN O., LINTERMANN B.: A modelling method and user interface for creating plants. In *Proceedings of Graphics Interface* (1997), pp. 189–197. 3

[Gre89] GREENE N.: Voxel space automata: modeling with stochastic growth processes in voxel space. *Computer Graphics 23*, 4 (1989), 175–184. 3

[Hal89] HALL R. W.: Fast parallel thinning algorithms: parallel speed and connectivity preservation. *Communications of the ACM 32* (1989), 124–131. 5

[Hon71] HONDA H.: 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 31* (1971), 331–338. 2

[IMT99] IGARASHI T., MATSUOKA S., TANAKA H.: Teddy: A sketching interface for 3D freeform design. In *Proceedings of SIGGRAPH* (1999), pp. 409–416. 5

[IOI06a] IJIRI T., OWADA S., IGARASHI T.: Seamless integration of initial sketching and subsequent detail editing in flower modeling. *Computer Graphics Forum 25*, 3 (2006), 617–624. 3

[IOI06b] IJIRI T., OWADA S., IGARASHI T.: The sketch L-system: Global control of tree modeling using free-form strokes. In *Proceedings of Smart Graphics* (2006), pp. 138–146. 2, 3

[IOOI05] IJIRI T., OWADA S., OKABE M., IGARASHI T.: Floral diagrams and inflorescences: Interactive flower modeling using botanical structural constraints. *ACM Transactions on Graphics 24*, 3 (2005), 720–726. 2, 3

[KP03] KARWOWSKI R., PRUSINKIEWICZ P.: Design and implementation of the L+C modeling language. *Electronic Notes in Theoretical Computer Science 86*, 2 (2003), 134–152. 2

[LD99] LINTERMANN B., DEUSSEN O.: Interactive modeling of plants. *IEEE Computer Graphics and Applications 19*, 1 (1999), 56–65. 2

[LPC*11] LIVNY Y., PIRK S., CHENG Z., YAN F., DEUSSEN O., COHEN-OR D., CHEN B.: Texture-lobes for tree modeling. *ACM Transactions on Graphics 30* (2011), 53:1–53:10. 3

[Mac83] MACDONALD N.: *Trees and networks in biological models.* J. Wiley & Sons, 1983. 8, 9

[MP96] MĚCH R., PRUSINKIEWICZ P.: Visual models of plants interacting with their environment. In *Proceedings of SIGGRAPH* (1996), pp. 397–410. 3

[NFD07] NEUBERT B., FRANKEN T., DEUSSEN O.: Approximate image-based tree modeling using particle flows. *ACM Transactions on Graphics 26*, 3 (2007), 88:1–88:8. 3

[OMKK06] ONISHI K., MARUKAMI N., KITAMURA Y., KISHINO F.: Modeling of trees with interactive L-system and 3D gestures. In *Biologically Inspired Approaches to Advanced Information Technology* (2006), pp. 222–235. 3

[OOI05] OKABE M., OWADA S., IGARASHI T.: Interactive design of botanical trees using freehand sketches and example-based editing. *Computer Graphics Forum 24*, 3 (2005). 3

[Opp86] OPPENHEIMER P.: Real time design and animation of fractal plants and trees. *Computer Graphics 20*, 4 (1986), 55–64. 2

[PBBPS99] POWER J., BERNHEIM-BRUSH A. J., PRUSINKIEWICZ P., SALESIN D.: Interactive arrangement of botanical L-system models. In *Proceedings of the ACM Symposium on Interactive 3D Graphics* (1999), pp. 175–182. 3, 8

[PHL*09] PAŁUBICKI W., HOREL K., LONGAY S., RUNIONS A., LANE B., MĚCH R., PRUSINKIEWICZ P.: Self-organizing tree models for image synthesis. *ACM Transactions on Graphics 28*, 3 (2009), 58:1–58:10. 3, 4, 5, 6, 10

[PJM94] PRUSINKIEWICZ P., JAMES M., MĚCH R.: Synthetic topiary. In *Proceedings of SIGGRAPH* (1994), pp. 351–358. 2

[PL90] PRUSINKIEWICZ P., LINDENMAYER A.: *The algorithmic beauty of plants*. Springer-Verlag, New York, 1990. With J. S. Hanan, F. D. Fracchia, D. R. Fowler, M. J. M. de Boer, and L. Mercer. 2

[PMKL01] PRUSINKIEWICZ P., MÜNDERMANN L., KARWOWSKI R., LANE B.: The use of positional information in the modeling of plants. In *Proceedings of SIGGRAPH* (2001), pp. 289–300. 2, 3

[RB85] REEVES W. T., BLAU R.: Approximate and probabilistic algorithms for shading and rendering structured particle systems. *Computer Graphics 19*, 3 (1985), 313–322. 3

[RCSL03] RODKAEW Y., CHONGSTITVATANA P., SIRIPANT S., LURSINSAP C.: Particle systems for plant modeling. In *Plant growth modeling and applications. Proceedings of PMA03*, Hu B.-G., Jaeger M., (Eds.). Tsinghua University Press and Springer, Beijing, 2003, pp. 210–217. 3

[RFL*05] RUNIONS A., FUHRER M., LANE B., FEDERL P., ROLLAND-LAGAN A.-G., PRUSINKIEWICZ P.: Modeling and visualization of leaf venation patterns. *ACM Transactions on Graphics 24*, 3 (2005), 702–711. 5

[RLP07] RUNIONS A., LANE B., PRUSINKIEWICZ P.: Modeling trees with a space colonization algorithm. In *Eurographics Workshop on Natural Phenomena* (2007), pp. 63–70. 3, 4, 5

[Sac04] SACHS T.: Self-organization of tree form: A model for complex social systems. *Journal of Theoretical Biology 230* (2004), 197–202. 3

[SLS*10] SHEK A., LACEWELL D., SELLE A., TEECE D., THOMPSON T.: Art-directing Disney's *Tangled* procedural trees. *ACM SIGGRAPH* (2010), 53:1. 1

[SN95] SACHS T., NOVOPLANSKY A.: Tree form: Architectural models do not suffice. *Israel Journal of Plant Sciences 43* (1995), 203–212. 3

[TLL*11] TALTON J., LOU Y., LESSER S., DUKE J., MĚCH R., KOLTUN V.: Metropolis procedural modeling. *ACM Transactions on Graphics 30*, 2 (2011), 11:1–11:14. 2

[TZW*07] TAN P., ZENG G., WANG J., KANG S.-B., QUAN L.: Image-based tree modeling. *ACM Transactions on Graphics 26* (2007), 87:1–87:7. 3

[Ula62] ULAM S.: On some mathematical properties connected with patterns of growth of figures. *Proceedings of Symposia on Applied Mathematics 14* (1962), 215–224. 3

[War09] WARD H. M.: *Trees. Volume V: Form and habit.* Cambridge University Press, Cambridge, 1909. 3

[WBCG09] WITHER J., BOUDON F., CANI M.-P., GODIN C.: Structure from silhouettes: a new paradigm for fast sketch-based design of trees. *Computer Graphics Forum 28*, 2 (2009), 541–550. 3

[WP95] WEBER J., PENN J.: Creation and rendering of realistic trees. In *Proceedings of SIGGRAPH* (1995), pp. 119–128. 3

[ZS07] ZAKARIA M., SHUKRI S.: A sketch-and-spray interface for modeling trees. In *Proceedings of Smart Graphics* (2007), pp. 23–35. 3