The botanical beauty of random binary trees

Luc Devroye[†] and Paul Kruszewski[‡]

School of Computer Science, McGill University
3480 University Street, Montreal, Canada H3A 2A7

ABSTRACT. We present a simple mechanism for quickly rendering computer images of botanical trees based on random binary trees commonly found in computer science. That is, we visualize abstract binary trees as botanical ones. We generate random binary trees by splitting based upon the beta distribution, and obtain the standard binary search trees as a special case. We draw them in PostScript to resemble actual botanical trees found in nature. Through flexible parameterization and extensive randomization, we can produce a rich collection of images.

KEYWORDS AND PHRASES. Tree drawing, tree simulation, tree visualization, beta distribution, random binary trees, PostScript.

Introduction.

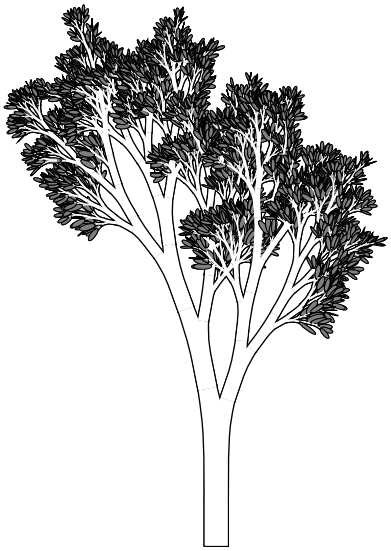


Figure 1. A visualization of a random binary tree with 5000 internal nodes.

[†] Research supported by NSERC Grant A3456 and FCAR Grant 90-ER-0291. Email: luc@cs.mcgill.ca.

[‡] Research supported by a 1967 NSERC Postgraduate Scholarship. Email: kruz@cs.mcgill.ca.

The computer imagery of realistic-looking trees has many applications ranging from the verification of botanical models to computerized landscaping and animation. In their book, *The Algorithmic Beauty of Plants*, PRUSINKIEWICZ AND LINDENMAYER (1990) provide an excellent overview of this emerging field. Through beautiful pictures, they and others have shown the mathematical elegance underlying simple biological systems. In this note, we hope to outline how computer data structures such as binary trees may be visualized in a similarly elegant way as botanical trees.

Suffix trees and tries are commonly used for storing text files for string searching (STEPHEN (1994)). When shown as a drawing in a window, a lot of information is revealed about the authorship, language, and nature of the text. Drawings can be used as simple, elegant signatures of files.

For example, the drawing in Figure 1 does not originate in nature. Rather this image has been created by visualizing a random binary tree in such a way as to resemble a botanical tree found in nature, i.e., each internal node of the random binary tree is drawn as a branch in the botanical tree and each external node is drawn as a leaf.

Our algorithm.

Our algorithm builds on the approach taken in KRUSZEWSKI (1994) which in turn is inspired by VIENNOT, EYROLLES, JANEY, AND ARQUÈS (1989) (see also VIENNOT (1990) or ALONSO AND SCHOTT (1995)). Indeed, we are heavily indebted to these authors for the idea of using combinatorial trees as a basis for drawing botanical ones. Logically speaking, we first generate a random binary tree by random splits and then we draw a corresponding botanical tree according to the resulting structure of the binary tree and to various controlling parameters (→). In practice, we generate the binary tree and draw a corresponding botanical one "on the fly", branch by branch, one after the other, in a preorder traversal. That is, for each subtree rooted at node u with children v and w, we draw the branch corresponding to u and then recursively draw the branches corresponding to nodes v and w. Our algorithm is implemented in PostScript and as such the algorithm runs *entirely inside* the printer.

Overall structure.

We generate the tree by random splits. It is well-known (e.g., DEVROYE (1994)) that many binary tree data structures such as binary search trees, tries, and PATRICIAS can be simulated by recursive random splits. That is, starting at the root with n nodes, let X be a $[0,1]$-valued random variable. Assign the left and right subtrees $\lfloor nX \rfloor$ and $n-1-\lfloor nX \rfloor$ nodes respectively. This splitting continues with independent identically distributed copies of X on the left and right subtrees until they each have only one node. Such a tree is called a random split tree. In our simulations, X is a beta random variable, and we call the resulting trees random beta trees.

After each node is created, its corresponding branch is drawn as a deformed rectangle. For each node u with children v and w, we determine for its corresponding branch, its length, width and branching angle. Implicit to our drawing

style is the idea of sap flow through the tree. That is, for each branch, the number of leaves in its subtree is supposed to be the key influence on its growth and there is some relation between size of the logical subtree and layout of the physical branch. Typically for example, the more leaves a branch has above it, the longer and wider that branch is.

Obviously, we are not the first to make such observations. The earliest reference which we could find are from about 1513 by Leonardo Da Vinci (RICHTER (1970)). In his book, *Botany for Painters*, Da Vinci sets up rules to guide artists in representing trees. Although Da Vinci attemps to give scientific explanations why things look as they do, his observations are first and foremost concerned with how things *should* look. We re-iterate that this is also our approach. That is, we are concerned with developing a model which produces convincing synthetic images rather than actually articulating how nature works.

Both VIENNOT ET AL. (1989) and KRUSZEWSKI (1994) use the HORTON-STRAHLER number of a node as the basis for functions of length, width and branching angles. At present, we prefer using subtree sizes. Typically, the width and length of a branch is a nondecreasing function f of $|u|$, the size of the subtree rooted at node u. Often, the æsthetically most pleasing results for the length and the width functions occur when the functions are of the form $c \ln |u|$ or $c\sqrt{|u|}$ where $c > 0$ is a constant (e.g., see Figure 2).

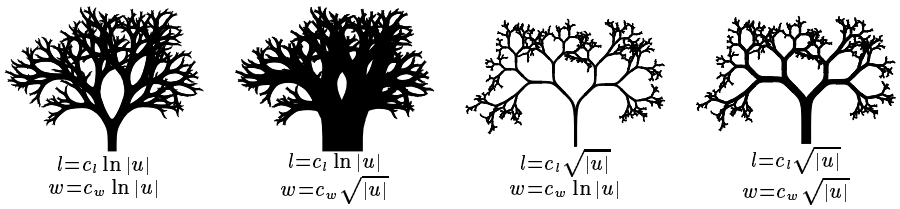


Figure 2. Various length and width functions for the same tree.

However, as Figure 3 shows, many other length functions may be used, such as $\frac{c'}{c+d}$, $\frac{c'}{c+d^2}$ or $\frac{c'}{c+\ln d}$, where d is the depth of the corresponding node and both c and c' are constants.

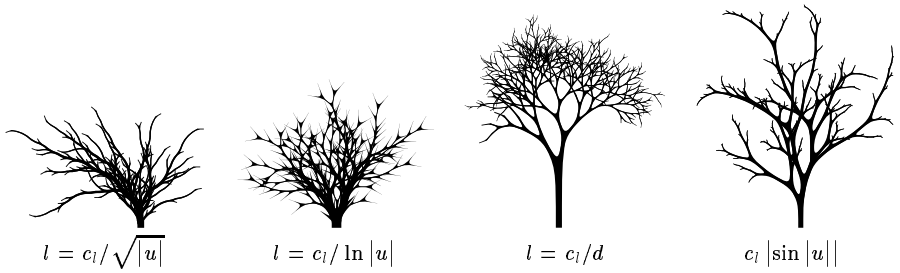


Figure 3. Examples of trees with atypical length functions.

Given two sibling branches v and w, it is often the case in nature that the larger branch deviates less in angle from the parent branch u than its sibling.

VIENNOT ET AL. (1989) determine three cases and corresponding angles: a branch is the main branch (typically 10°), secondary branch (typically 25°), or a fork branch where it and its sibling are about equal in value so that both angles are also about equal (typically 30°). Classification of branches is based the HORTON-STRAHLER number. However, since many families of random binary trees have logarithmic HORTON-STRAHLER numbers in the number of nodes (see e.g., DEVROYE AND KRUSZEWSKI (1994,1995)), similar comparisons such as

$$\theta_v = \begin{cases} 10°, & \text{if } \lfloor \log |v| \rfloor > \lfloor \log |w| \rfloor, \\ 25°, & \text{if } \lfloor \log |v| \rfloor < \lfloor \log |w| \rfloor, \\ 30°, & \text{if } \lfloor \log |v| \rfloor = \lfloor \log |w| \rfloor, \end{cases}$$

are equally acceptable (e.g., see the first drawing in Figure 4). Nonetheless, the possibilities for branching angles are endless. In this figure, each tree has the same number of nodes and is drawn from the same probability distribution. The second drawing uses sibling sizes directly to determine angle, i.e., $\theta_v = \frac{|w|}{|v|+|w|} \times 30°$. Finally, the last two drawings rely on depth d of the branch in the tree, i.e., $\theta = \frac{27°}{d+1}$ and $\theta = \frac{23°}{\ln d + 1}$.

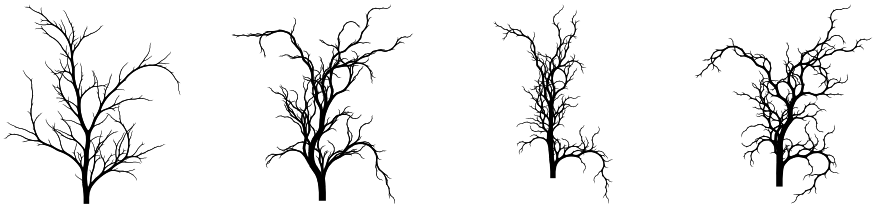


Figure 4. Various angle functions for the same tree.

For greater realism, each angle θ may be multiplied with $\cos(2\pi U)$ where U is uniform$[0,1]$ to simulate projection in the plane of a random 3-d rotation (e.g., Figure 5 shows this effect on the second drawing in the previous figure). In all cases, to avoid any absurdly asymmetric drawings, at each split, we flip a coin and place θ by $-\theta$ with probability $1/2$.

Figure 5. Simulated random 3-d angles.

Drawing polygonal branches typically results in rough-looking notches where the branches meet. VIENNOT ET AL. (1989) fill in these joints with

small triangles. We avoid this problem by drawing smooth, rounded forks  rather than individual branches (cf. BLOOMENTHAL (1985)).

That is, the "buds" of the left and right child branches are also drawn with the parent as one smooth unit. We then overlay the buds with the corresponding child forks for a smooth fit. This means that in practice, the size and orientation of a branch is worked out when its parent branch is drawn. Thus, the child forks are laid over the parent fork . Note that all of the curves are created by the PostScript command `curveto` which implements Bézier curves (see p. 140 in ADOBE (1985)). This layering continues throughout the preorder traversal.

Beta distribution.
We split according to the beta distribution as it yields a rich family of branching patterns. The beta(a,b) has density

$$f(x) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\, x^{a-1}(1-x)^{b-1}, \qquad 0 < x < 1,$$

where $a, b > 0$ are parameters and Γ is the gamma function. For example, beta(1,1) produces random binary search trees. Beta trees are defined in DEVROYE (1986) as trees in which the sizes of right and left subtrees are multinomial $(n, X, 1-X)$ where X is as before. The multinomial beta trees are slightly different from the model used for tree drawing, but the differences are so minor that for tree drawing purposes, we prefer to use the $(\lfloor nX \rfloor, \lfloor (1-X)n \rfloor)$ model of this paper.

For the multinomial beta trees pruned as soon as a subtree size reaches one, if a,b tend to infinity such that $\frac{a}{(a+b)} = p$, then one obtains a trie with symbol probabilities p and $1-p$ (see e.g., PITTEL (1985)). The beta distribution is versatile primarily because varying a and b results in a wide family of trees with logarithmic average depth and height. That is, the bushiness and elongation of the trees can be controlled by varying the parameters. More formally DEVROYE (1995) shows the following theorems for random split trees in general.

THEOREM 1. *Let D_n be the depth of the last node in a random split tree with n nodes. Then*

$$\frac{D_n}{\log n} \to \frac{1}{\mu} \qquad \textit{in probability as } n \to \infty,$$

and $\mathbf{E}\{D_n\}/\log n$ tends to the same limit, where $\mu = 2\mathbf{E}\{Y \log(1/Y)\}$, $Y \in [0,1]$ is X and $1-X$ with equal probability, and X is the branch-splitting random variable introduced earlier.

THEOREM 2. *Let H_n be the height of a random split tree with n nodes. Then*

$$\frac{H_n}{\log n} \to \gamma \qquad \textit{in probability as } n \to \infty,$$

where $\gamma = \inf\{c : e^{t^*}(2m(t^*))^c < 1\}$, $m(t) = \mathbf{E}\{Y^t\}$, $t \geq 0$, t^* is the unique solution of $m'(t)/m(t) = -1/c$, and Y is as in Theorem 1.

With random beta trees, one can choose the desired expected depth and height and solve the above formulas to determine explicit values for a and b. Note, for example, that $1/\mu$ can take any value between $1/\log 2$ and ∞. We implement this distribution by the PostScript uniform random number generator `rand` and Cheng's method for beta variates (CHENG (1978) as explained on p. 438 of DEVROYE (1986)).

Figures 6 and 7 show the flexible nature of the beta distribution. Each tree consists of 500 nodes and is drawn using the same rules (for length, width, and angle) but for different beta parameters. As Figure 6 shows, as $a \to \infty$, and $b = a$, the splitting is even and deterministic, and as $a \to 0$, $b = a$, the splitting is asymmetric and unstable.

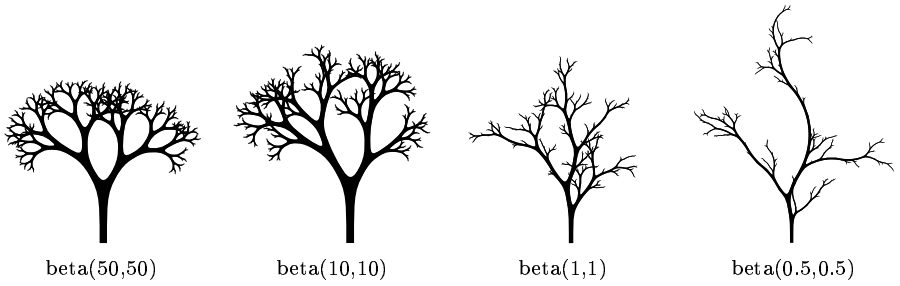


Figure 6. Examples of random beta(a,a) trees with 500 nodes.

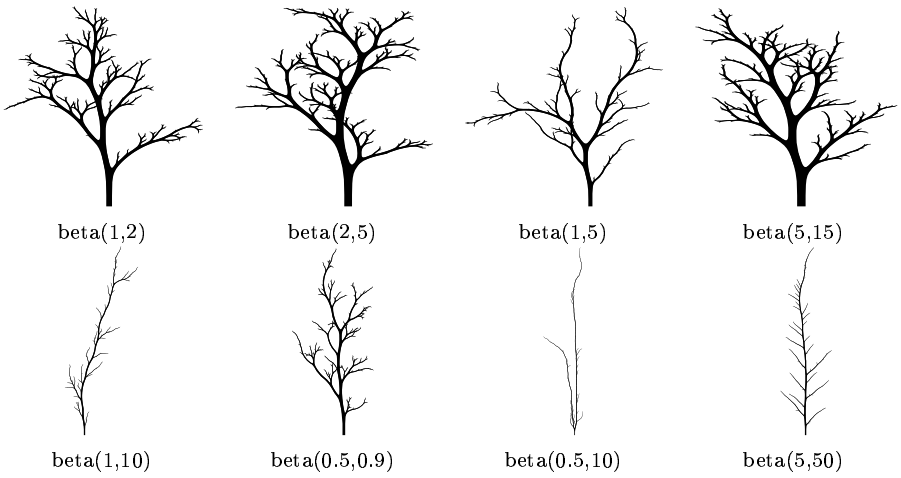


Figure 7. Examples of random beta(a,b) trees with 500 nodes.

Randomization.
Our algorithm is heavily randomized. The underlying tree structure is generated by random splits. Furthermore, all functions such as length, width and branching angle can be perturbed using randomness. Angles can be randomized based upon subtree sizes, depths and split ratios, for example.

Leaves.
Realistic-looking leaves are an important component for any tree drawing program. We use a very simple rectangular shape based on the examples found in SUGDEN (1984). A leaf consists of an apex (top) and a base. Shape is controlled by varying the apex height, base depth and leaf width. As Figures 8 and 9 show, we have nine different apices and four different bases, each constructed with simple Bézier curves.

Figure 8. Various leaf apices.

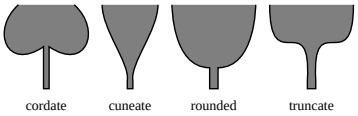


Figure 9. Various leaf bases.

Added realism can be achieved by drawing two-dimensional projections of the leaves. Rather than actually modelling in three dimensions, sufficient realism can be achieved by rotating and projecting the leaves.

Various tropisms: sun and wind.
Tropism is the property by which an organism turns in a certain direction in response to external stimulus. In plants, this stimulus is primarily the sun and hence heliotropism has been incorporated into many models (e.g., CHIBA, OHKAWA, MURAOKA, AND MIURA (1994)). We simulate heliotropism according to sun position and intensity. With respect to intensity, we use the admittedly naïve idea that the larger the branch the more light it receives over its lifetime and thus the more it reacts by changing its angle. That is, for node u after θ_u is determined, θ_u is multiplied by an intensity factor (based on $|u|$) which pulls branch u closer to the sun. In Figure 10, we take the beta(1,5) tree with 500 nodes from Figure 7 and subject it to increasing sun intensity with the sun directly overhead. However as Figure 10 shows, we neglect to consider that leaves tend to spread out to maximize coverage.

Wind is also an important environmental factor. Both VIENNOT ET AL. (1989) and KRUSZEWSKI (1994) simulate wind by changing the underlying structure; the former always flips larger branch to one side while the later

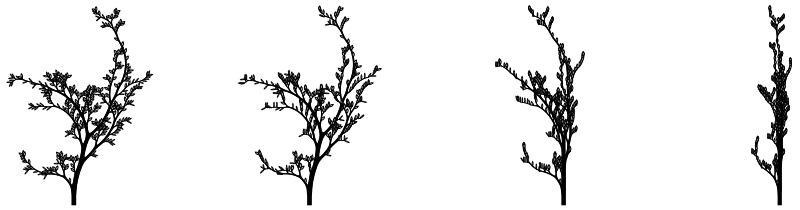


Figure 10. A tree under increasingly intense sun.

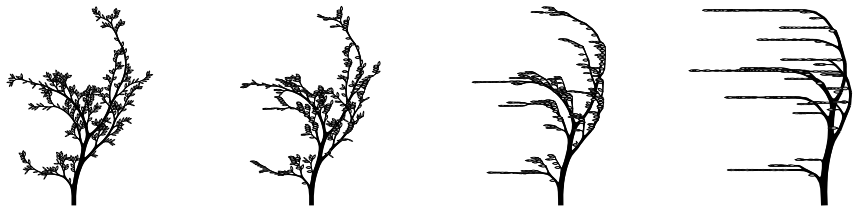


Figure 11. A tree under increasingly intense wind.

uses asymmetric tries. As Figure 11 shows, by placing the sun perpendicular to the ground and inverting the intensity function (i.e., larger branches should bend less than smaller ones), reasonable wind can be simulated.

Finally, if we set the wind to blow from above, we can simulate the effect of droughts or flexible branches such as those found in weeping willows.

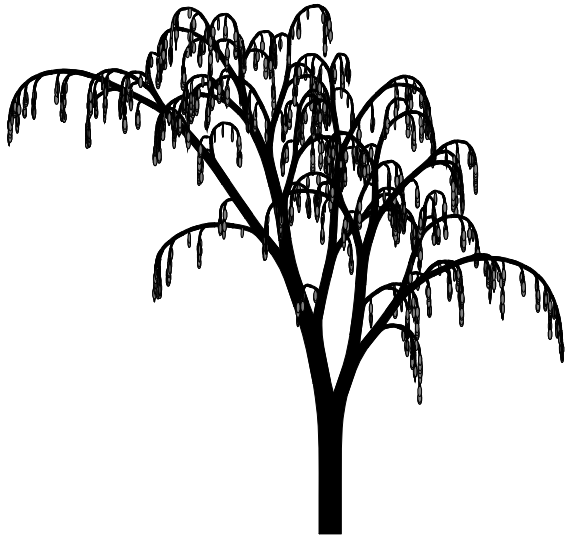


Figure 12. A weeping willow.

Three-dimensional drawing.
Botanical trees are three-dimensional objects. Therefore, added realism is obtained by drawing the trees in three dimensions and projecting them onto the

two-dimensional plane. In the 3-d case, we now consider the branching angle from the 2-d case to be a rotation about the z-axis and add a second rotation about the z-axis. This approach was first taken by ANON AND KUNII (1984). We have forking, main and secondary angles of sizes 30°, 70°, and 20° respectively. Branches are now cylindrical and smooth Bézier curves require a serious computational effort. Currently, we opt for the simpler solution of representing the branches as solid cylinders. This approach is very acceptable when the branches are very thin (e.g., Figure 13). However, drawings of thick-branched trees are rather unappealing. We are currently working on a new 3-d model which will produce the same smooth forks as in the 2-d model.

Figure 13. A three-dimensional tree.

Conclusions.

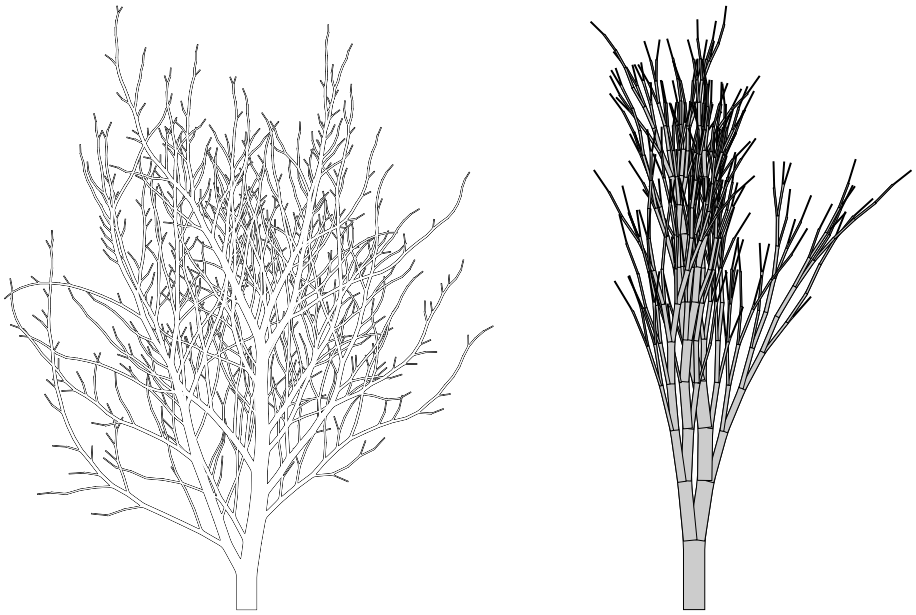
All of the images were generated in PostScript on a 600 dpi Apple LaserWriter Pro with 8 megabytes of memory. All files[†] are completely self-contained and are about 27 kilobytes long, of which more than half is documentation. For example, Figure 1 has 5000 branches and takes approximately 33 minutes to print. Image rendering by `ghostview` is about eight times faster.

Many extensions and enhancements can be imagined. Probably, the most desired would be to wrap the program in a graphical user-interface. Currently, different trees are generated by modifying the PostScript code by hand and then re-viewing with the PostScript previewer `ghostview`. A graphical interface would allow the user to freely change parameters and then instantly view resulting changes. We do not grow the tree dynamically to model physical growth. However, if we re-draw the tree after each successive node is added, we could have a reasonable animation of tree growth. Finally, not all trees are binary, we hope to extend our model to arbitrary k-ary trees.

[†] Our programs are available by anonymous `ftp` at `ftp.cs.mcgill.ca` in the directory `pub/tech-reports/library/code/botan-ical.trees/`.

Acknowledgements.

We thank Sue Whitesides for her simulating conversations, useful advice and ongoing encouragement.



A birch tree from Quebec's Laurentians. A yucca tree.

Figure 14. Sundry trees.

References.

ADOBE SYSTEMS INC. (1985). *PostScript Language Reference Manual.* Reading, MA: Addison-Wesley.

ALONSO, L. AND R. SCHOTT (1995). *Random Generation of Trees.* Dordrecht, The Netherlands: Kluwer Academic Publishers.

ANON, M. AND T. KUNII (1984). Botantical tree image generation. *IEEE Computer Graphics Applications 4*, 10–34.

BLOOMENTHAL, J. (1985). Modeling the Mighty Maple. In *Proceedings of SIGGRAPH'85, Computer Graphics*, Volume 19, pp. 305–311.

CHENG, R. C. H. (1978). Generating beta variates with nonintegral shape parameters. *Communications of the ACM 21*, 317–322.

CHIBA, N., S. OHKAWA, K. MURAOKA, AND M. MIURA (1994). Visual simulation of botanical trees based on virtual heliotropism and dormancy break. *Journal of Visualization and Computer Animation 5*(1), 3–15.

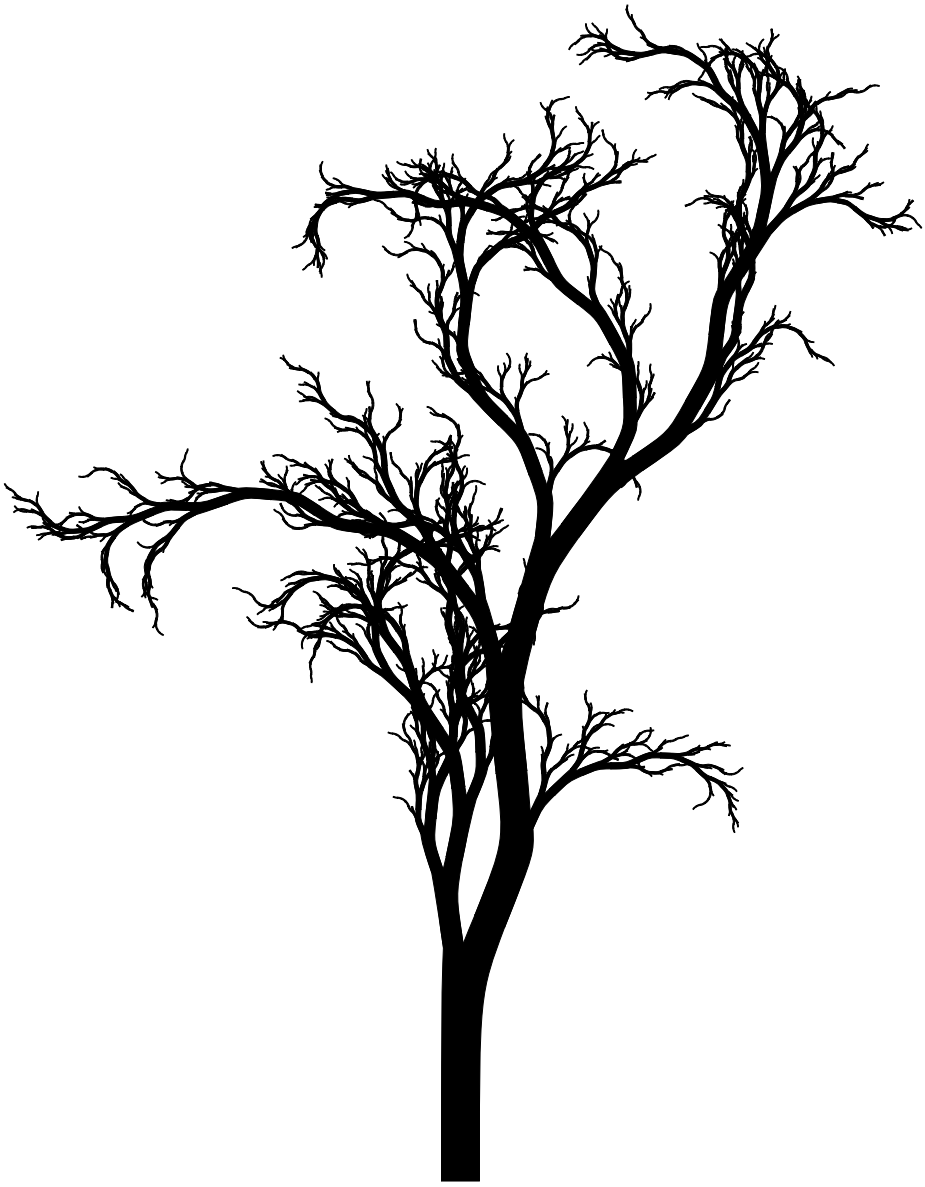


Figure 15. Passau im Herbst.

DEVROYE, L. (1986). *Non-Uniform Random Variate Generation.* New York: Springer-Verlag.

DEVROYE, L. (1994). *Lectures Notes for Computer Science 690A—Probabilistic Analysis of Algorithms and Data Structures.* Montreal: School of Computer Science, McGill University.

DEVROYE, L. (1995). Universal limit laws for depths in random trees. (Sub-

mitted).

DEVROYE, L. AND P. KRUSZEWSKI (1994). A note on the HORTON-STRAHLER number for random trees. *Information Processing Letters 52*, 155–159.

DEVROYE, L. AND P. KRUSZEWSKI (1995). On the HORTON-STRAHLER number for random tries. (Submitted).

KRUSZEWSKI, P. (1994). Using the HORTON-STRAHLER number to draw trees. Technical Report SOCS-94.1, School of Computer Science, McGill University.

PITTEL, B. (1985). Asymptotical growth of a class of random trees. *Annals of Probability 13*, 414–427.

PRUSINKIEWICZ, P. AND A. LINDENMAYER (1990). *The Algorithmic Beauty of Plants*. New York: Springer-Verlag.

RICHTER, J. P. (1970). *The literary works of Leonardo Da Vinci* (3 ed.). New York: Phaidon Publishers Inc.

STEPHEN, G. A. (1994). *String Searching Algorithms*. Lecture Notes Series on Computing. New York: Springer-Verlag.

SUGDEN, A. (1984). *Longman Illustrated Dictionary of Botany*. Essex, UK: Longman Group Limited.

VIENNOT, X. G. (1990). Trees everywhere. In A. Arnold (Ed.), *Proceedings of the 15th Colloquium on Trees in Algebra and Programming, Copenhagen, Denmark, May 15-18, 1990, Lecture Notes in Computer Science*, Volume 431, Berlin, pp. 18–41. Springer-Verlag.

VIENNOT, X. G., G. EYROLLES, N. JANEY, AND D. ARQUÈS (1989). Combinatorial analysis of ramified patterns and computer imagery of trees. In *Proceedings of SIGGRAPH'89, Computer Graphics*, Volume 23, pp. 31–40.