THE UNIVERSITY OF CALGARY

Interactive Procedural Modelling of Trees and Landscapes

by

Steven Longay

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF Ph.D. IN COMPUTER SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

CALGARY, ALBERTA

September, 2014

© Steven Longay 2014

THE UNIVERSITY OF CALGARY

FACULTY OF GRADUATE STUDIES

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance, a thesis entitled "Interactive Procedural Modelling of Trees and Landscapes" submitted by Steven Longay in partial fulfillment of the requirements for the degree of Ph.D. in Computer Science

Supervisor
Dr. Przemyslaw Prusinkiewicz
Department of Computer Science

Dr. Richard Levy
Faculty of Environmental Design

Dr. Faramarz Samavati
Department of Computer Science

Dr. Pierre Poulin
University of Montreal

Dr. Anthony Tang
Department of Computer Science

Date

Abstract

Virtual trees are an essential component of many computer generated outdoor scenes. Creation of digital tree models is a difficult and time consuming process requiring specially trained artists and therefore trees are often selected from a library of models. These pre-generated models are expensive, of limited quality and do not suffice when a particular artistic design is required. Moreover, when multiple tree models are combined into scenes, they lack important characteristics of trees that were grown together. My thesis focuses on the development of algorithms and modelling software capable of generating diverse tree forms that are botanically plausible yet easily controlled to meet artistic requirements. I propose a new model of tree growth which enhances the directability and diversity of previous algorithms. A novel multi-touch tablet interface is presented which allows for the free form creation of trees through brushing, pruning and bending of branches into the desired form. Furthermore, I present an algorithm and interface for interactively designing an entire landscape of trees. Neighboring trees exhibit characteristics of being grown together yet they are simulated independently, taking advantage of parallel computing techniques. This ability to easily generate realistic trees and landscapes has applications in the fields of computer graphics and animation, digital landscape design in architectural visualization and computer games.

Acknowledgments

I would like to thank my supervisor Dr. Przemyslaw Prusinkiewicz for introducing me to the field of procedural modelling and providing me freedom in pursuing this project. Above all I am grateful for your enthusiasm and always striving for excellence. Thank-you to all the members of the lab, especially Adam Runions, for your extensive comments and testing of TreeSketch. Last but not least, I am grateful for the financial support from the Graphics, Animation and New Media Network of Centers of Excellence (GRAND).

Table of Contents

List of Figures

Chapter 1

Introduction

Due to the complexity of tree structures, the creation of trees for computer imagery requires specialized modelling methods. Procedural modelling generates content by means of a procedure or program [Ebert et al., 2003] and is commonly used as a tool that manages the complexity of generating digital trees. In practice, tree models often have strict artistic requirements and are required to mimic characteristics existing real-world trees or concept sketches. Previous attempts at integrating user control with procedural tree models have failed to capture a wide diversity of realistic tree models and often require specialized programming or biological knowledge. The crux of the problem is the inherent tension between procedural model generation and user control. The goal of a procedural model is to generate complex content automatically. Yet, control over every aspect of the resulting structure is desired.

The ability to easily generate complex tree forms is important for scene dressing of computer generated imagery, but also has prospective applications in education, horticulture and landscape design. The models in this thesis have a sound biological foundation and they can therefore be used to develop further understanding of tree development and enhance appreciation of nature.

My hypothesis is that procedural models can be developed for the creation of trees and landscapes which biologically motivated and can be controlled at interactive rates without requiring the modeller to have a computer science or biological background. Such models

should provide the user with high level controls that mimic functionality they are already familiar with, such as painting, bending of branches and pruning.

In his acceptance speech for the 2009 Steven Coons Award, Rob Cook repeatedly stressed the importance of interactive techniques to the future of computer graphics. Specifically, he advocated intuitive 3D editing, concluding that “our tools need to feel more like brushes" and “programs should have an understanding of what you are drawing and use this to give a higher level control" of the content. My work aims to address these objectives in the field of tree modelling.

1.1 Methodology

Current methods for generating tree structures follow two main paradigms: procedural tree generation and reconstruction of tree geometry from photographs or laser scans. The reconstruction methods can potentially provide faithful three-dimensional geometry of real trees with little modeller 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 accentuated for trees that only grow in forests, and compounded by the limited geographic range in which some trees grow. Procedural models potentially do not suffer from these limitations. Nevertheless, the development of methods capable of generating widely diverse trees, botanically plausible yet easily controlled to meet artistic requirements, has remained a challenge. Thus, creators of procedural tree models strive to integrate user control into their methods.

One approach to integrate user control into procedural models is by solving the inverse problem: the modeller specifies the approximate final structure (e.g. sketch or photograph)

and the system solves for the best-matching parameters of a procedural model to generate the structure [Talton et al., 2011]. This concept, inverse procedural modeling, has been used to create tree models similar to an exemplar but adapted to fit environmental constraints [Šťava et al., 2014]. While these systems have the potential to produce high quality results, they do not run at interactive rates and the computation of a single tree can take hours.

Sketch based interfaces have been proposed as a technique to introduce artistic control while expediting the process of procedural tree modelling [Wither et al., 2009, Chen et al., 2008, Okabe et al., 2005]. Due to the complexity of tree architectures, artistic sketches are not detailed enough to produce visually realistic models directly. Consequently, tree modelling systems rely on recursive algorithms or a database of sample trees to generate the finer levels of detail. This can result in trees which look repetitive and unnatural. Procedural models based on competition for space [Runions et al., 2007] or light [Pałubicki et al., 2009, Pałubicki, 2012] generate more realistic trees that do not suffer from these artifacts. These methods, however, do not focus on the interactive control of generated models.

The radius associated with a brush provides an inherent advantage over sketches because the modeller is specifying their input over a larger area. This provides more freedom to the procedural model giving it the ability to produce a superior result. As the radius of the brush tends to zero, the interface becomes sketch based, thus there is no loss of control.

Simulating tree growth of an entire landscape is computationally expensive. Moreover, since the trees are grown together, changes to an individual tree may require re-computing the entire landscape. Therefore, simulation of landscapes is usually partitioned into multiple levels making it more manageable [Deussen et al., 1998]. Lane and Prusinkiewicz [2002] propose a system in which the distribution of tree bases is first computed by simulating interactions between tree species resulting in clustering and inhibition. Using this distribu-

tion, tree models representative of each species are *instanced* at the location of each tree. Independently rotating and adjusting the size of each model helps to provide the illusion of diversity when viewed from a distance, but is not sufficient when the viewer is close. The main problem is the lack of variability in crown shapes. Real tree crowns are highly adaptive and dependent on neighboring trees.

This thesis describes methods for interactive procedural modeling of trees and landscapes. A core component is an algorithm for simulating tree growth which affords directable control over the generated tree form. This growth algorithm is implemented in TreeSketch, a multi-touch tablet application that presents an intuitive brush based interface while allowing for intuitive editing of the tree structure such as pruning and bending branches. Scaling from trees to forests, I present a multi-level technique for the interactive specification of entire landscapes. Resulting trees have the natural appearance of those that have been grown together, yet they can be simulated independently in parallel. The presented methods are capable of generating a wide range of diverse forms that are botanically plausible yet easily controlled to meet requirements.

1.2 Contributions

Interactive procedural modeling of trees and landscapes is an interdisciplinary subject area spanning the fields of computer graphics, biology, human-computer interaction and art. Existing methods and systems do not support the diversity, realism, artistic control and free-form modeling experience that my research aims to achieve. This section describes main scientific advancements and contributions I have made towards these goals. They can be split into three categories:

- **Growth Simulation**: Directability, realism and diversity. I present an algorithm for tree growth which adds directability to previous models. This allows for a diverse set of realistic and artistic tree forms to be created. The ability to create realistic trees that naturally grow into various shapes is essential to the algorithm for generating landscapes.

- **TreeSketch**: Interface, widgets and algorithmic support of interaction. I developed a multi-touch brush-based interface for free-form interactive design of trees. A set of interactive widgets provides intuitive control over correlated parameters. Algorithms for bending of branches and defining branch widths have been devised for the purpose of interactive tree modeling. An augmented-reality interface enables creation and visualization of trees within real world environments while allowing them to adapt to their surroundings.

- **Landscapes**: Multi-level specification, biologically motivated, scalable. I have developed an algorithm for generating spatial constraints on the shape of tree crowns from a distribution of tree bases. These constraints capture important effects resulting from the competition between neighboring trees. Each tree can be simulated independently, enabling the use of parallel processing techniques that greatly enhance the speed of generation.

1.3 Organization

Throughout this thesis I aim to present the biological concepts and observations first, followed by an exploration of the corresponding simulation techniques and finally interfaces that allow for artistic guidance and manipulation of the tree structures. I start by introducing

the form of trees from a visual and biological perspective (Chapter 2). After a review of previous work on tree modelling (Chapter 3), I present the model of tree growth that is used in the remainder of this thesis (Chapter 4). Chapter 5 presents TreeSketch, the interactive multi-touch application for modelling trees. Algorithms for the interactive design of landscapes are presented in Chapter 6. Finally, Chapter 7 contains a discussion of future work and concluding remarks.

The text from [Longay et al., 2012] has been incorporated throughout this thesis and adjusted where necessary to account for new results. Figures used from this work directly are cited in the captions. Section C.1.2 is based on the poster [Longay et al., 2010].

Chapter 2

Tree Morphology

Considering that trees have no central brain to orchestrate their development, it is wondrous to see the complex yet harmonious forms they are able to achieve. The potential of a tree is determined through its genetic information even before it becomes a seedling. Yet, the variability in form present between trees of the same species can be staggering. Much research has gone into determining how the branching structure of trees emerge.

When creating a procedural model to simulate tree development, it is important to identify which aspects are most important in determining the form of the tree. By parameterizing all aspects of tree form, the model would become as complicated as reality itself and no longer be of use [Prusinkiewicz, 1998]. The following section discusses characteristics of tree development which directly correspond to visual differences in tree form.

2.1 Key Discriminates of Form

A tree develops through the production of *metamers* by active *apical (terminal) buds*. A metamer consists of a branch segment called an *internode*, associated with one or more *lateral buds* subtended by a *leaf* (Figure 2.1). The sequence of internodes produced by a bud forms an *axis*. Lateral buds may give rise to new axes, which leads to the development of *branches*. The branch system of a tree, supported by the *trunk*, constitutes the tree *crown*.

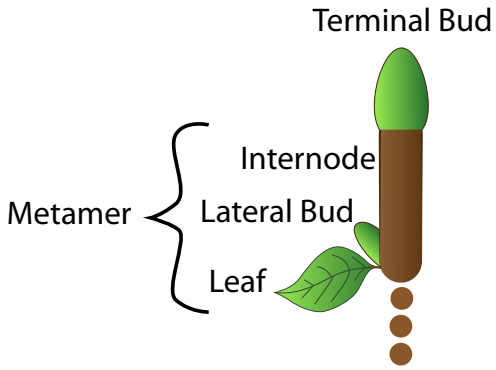


Figure 2.1: Tree composition

Traditional botanists have distinguished a number of features they found useful in describing tree forms [Ward, 1909]. The overall *silhouette* of a tree is usually characterized using descriptive terms such as *columnar*, *pyramidal*, or *broad*. Whether or not a tree has a well defined trunk has a dramatic impact on its visual form. In biology, such forms are termed *excurrent* and *decurrent* respectively. The presence of a trunk is thought to be caused by the phenomenon of *apical dominance*, where the terminal bud of a branch suppresses growth of its lateral branches.

Lateral branches are issued at a particular angle from their supporting axis, the *branching angle*. The path of a branch curves and changes throughout its development, yet many branches have a tendency to maintain a preferred orientation with respect to gravity, *gravitropism*. 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 [1995] quantified and unified these notions by introducing the *gravitropic set-point angle* (GSA): the angle θ to the direction of gravity that the branch axes tend to maintain (Figure 2.2).

Figure 2.2: Examples of graviropism in real trees. Left, a birch tree where the angle of tropism changes over time. Right, the gravitropic set-point angle. Images courtesy of Wojtek Pałubicki.

The related phenomenon of *gravimorphism* has been recognized by biologists as also having an important impact on tree form [Barthélémy and Caraglio, 2007]. Gravimorphism refers to the differential growth potential of buds that depends on the orientation of the bud with respect to its parent axis. In the case of *epitony*, buds on the upper side of their supporting axes are favoured. In contrast, *hypotony* refers to the favouring of buds on the underside of branches while *amphitony* favours buds that lie in horizontal positions (Figure 2.3).

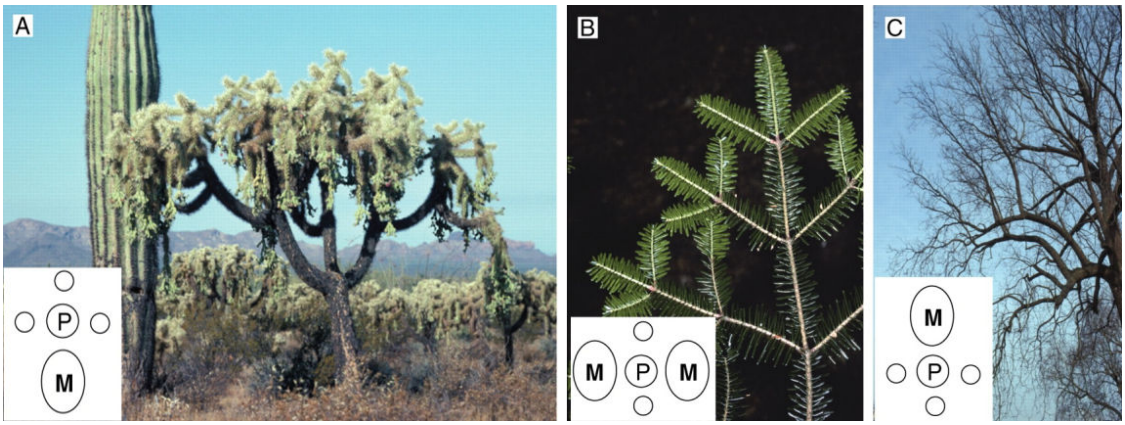


Figure 2.3: Image from [Barthélémy and Caraglio, 2007] showing examples of real trees with strong gravimorphism effects. *M* refers to *mesotony*, the privileged development of lateral branches on their parent axis *P*. A) hypotony, B) amphitony and C) epitony.

Harper [1977] identifies the *neighborhood effect* as one of the fundamental characteristics controlling tree form, relating the adaptive shape of tree crowns to the presence of neighbors. Also known as *crown plasticity* or the *river-bank effect* [Hallé et al., 1978], this can be seen most predominantly at the edge of the forest or on the bank of a river, where trees can be found leaning into open space and away from their neighbors (Figure 2.4). By displacing its crown away from its neighbors, the tree is able to reduce competition and obtain more resources [Longuetaud et al., 2008, 2013].

Figure 2.4: The neighborhood effect in a stand of trees. The tree on the right grows away from its neighbors and has a tilted trunk. Image from https://flic.kr/p/8J8jfW.

2.2 Developmental Paradigms

Hallé et al. [1978] proposed the *architectural models* of tree development. They identify twenty-three developmental templates (patterns) that capture the form of tropical trees through the process of reiteration. From this perspective, the genetic makeup of the tree species defines its developmental pattern and is the main controlling factor of tree form. The structure of the tree emerges by repetition of this predefined pattern, and the environment

of the tree is seen as having only a disrupting effect on this pattern. As noted by [Hallé et al., 1978], when looking at trees in nature, it can be difficult to discern the particular developmental pattern due to the plasticity of its expression. Consider the tree in Figure 2.5, no repetitive developmental pattern is evident and thus it is unclear what developmental pattern could be used to recreate it.

Figure 2.5: Image of a real tree. The overall form is well balanced but it is difficult to discern what the repetitive developmental pattern would be. Photograph courtesy of Przemyslaw Prusinkiewicz.

The architectural models of tree development apply to the form of trees in tropical climates where the environment is favorable to growth. However, trees that grow in temperate climate must be more adaptable and competitive in order to survive the harsher

environment [Hallé et al., 1978]. Plant development can also be viewed from a different perspective, where a plant is considered as a modular collection of semi-autonomous units [White, 1979, Harper, 1985]. This idea was refined by Sachs and his collaborators [Sachs et al., 1993, Sachs and Novoplansky, 1995, Sachs, 2004, 2006] who propose:

> "The form of a tree of a given species is generated by self-organization in which alternative branches inhibit or compete with one another, following no strict plan or pre-pattern" [Sachs, 2004] p. 200.

From this perspective, competition for space and resources between individual units of a plant is the key factor determining tree form. The genetic makeup of the tree may determine its capabilities, but the overall form emerges through competition. Considering the tree in Figure 2.5, the overall well-balanced form can be viewed as the emergent result of competition while more specific characteristics of the tree can be described qualitatively at a higher level. For example, this tree can be described as having a broad crown composed of branches tending in the horizontal direction (plagiotropism). New shoots are oriented upwards (negative gravitropism) and have a clear preference to grow from the upper side of their supporting axis (epitony).

In this thesis, I view the development of trees from the latter perspective, generating tree form by means of self-organization. While many of the discussed features are defined in fuzzy terms such as "tendencies" and "preferences", they correspond well to specific parameters of self-organizing tree models, as shown in subsequent chapters. It is worth noting that while these two perspectives are not mutually exclusive, the latter represents a paradigm shift from architecture to environment being the key determinant of form [Sachs and Novoplansky, 1995].

Chapter 3

Previous Work

Procedural modelling is widely used as a tool to manage the complexity of creating virtual objects. While trees have become the 'poster child' of procedural modelling, it has been used to generate a wide variety of objects such as terrain [Musgrave et al., 1989], buildings [Mueller et al., 2006], city layouts [Lipp et al., 2011] and even art [Talton et al., 2011]. Attractive attributes such as data amplification, compression and generation speed [Smith, 1984] are promised by procedural modelling. In practice, however, procedural models are notoriously hard to control and often small changes in parameters result in large, unintuitive changes in the resulting model. The goal of interactive procedural modelling is to improve this situation by providing the user with a more intuitive, higher level control over the resulting structure. This chapter reviews previous research in the field of interactive procedural modelling of trees and landscapes.

3.1 Procedural Tree Modelling

Algorithms for procedural tree modelling 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. This section reviews 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.

3.1.1 Recursive and Hierarchical Models

A natural extension from the study of tree architecture is procedural tree models based on a recursive or hierarchical branching pattern. The structure of the tree emerges by repetition of this predefined pattern. Most older algorithms [Honda, 1971, Aono and Kunii, 1984, Bloomenthal, 1985, Oppenheimer, 1986, Prusinkiewicz and Lindenmayer, 1990] 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 [de Reffye et al., 1988], but the modeller's ability to control them is impeded by the need to understand the intricacies of the generative process and their software implementation. These difficulties have been partially circumvented by limiting the modeller to manipulate a set of exposed parameters using standard control panels [Oppenheimer, 1986, Prusinkiewicz and Lindenmayer, 1990] or specialized graphical editors [Ijiri et al., 2005, 2006]. Additional techniques include confining modifications of the algorithm to the choice between predefined options [de Reffye et al., 1988], introducing dedicated modelling languages to specify the generative algorithms [Prusinkiewicz and Lindenmayer, 1990, Karwowski and Prusinkiewicz, 2003], and introducing graphical interfaces for composing the algorithm from predefined components [Lintermann and Deussen, 1999]. 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 modelling topiary trees [Prusinkiewicz et al., 1994], 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) [Talton et al., 2011].

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 [Prusinkiewicz et al., 2001]. 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 [Reeves and Blau, 1985]. Methods proposed to define silhouettes include specialized surfaces controlled by a small number of parameters [Reeves and Blau, 1985, Weber and Penn, 1995, Boudon et al., 2003] and arbitrary surfaces of revolution with a graphically defined profile [Deussen and Lintermann, 1997, Prusinkiewicz et al., 2001]. Boudon et al. [2003] 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. [2009] used a sketch-based interface to define 2D silhouettes of branches that were subsequently rearranged in 3D. Xu and Mould [2012] defined the global form of the tree by combining geometric primitives filled with randomly weighted volumetric graphs. Branches are then defined by least-cost paths through these graphs.

Control of limb shape

The top-down systems by Deussen and Lintermann [1997], Prusinkiewicz et al. [2001] and Boudon et al. [2003] allowed the modeller to specify the shape of selected axes using a

parametric curve editor. A more intuitive method was introduced by Okabe et al. [2005] and Ijiri et al. [2006] who defined the axes of two-dimensional branches strutted 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 [Ijiri et al., 2005], projections of sketches onto billboards arranged in 3D, and multiple projections [Ijiri et al., 2006]. Onishi et al. [2006] bypassed the problem of inferring 3D axes from 2D sketches by using a 3D input stylus. Complementing these techniques, Power et al. [1999] 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 [Okabe et al., 2005] or imposing a phyllotactic pattern of branch arrangement [Wither et al., 2009] (phyllotaxis is explained in detail in Chapter 4). In the technique introduced by Chen et al. [2008], a sketch representing main branches and, optionally, the desired silhouette of the tree, is compared with a database of reference trees. 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. [2007] inferred parameters used for branch proliferation from the layout of visible branches. Livny et al. [2011] proliferated branches by rearranging templates generated with L-Systems to fill given envelopes.

3.1.2 Self-Organizing Trees

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 modeller. 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 [White, 1979, Harper, 1985, Sachs and Novoplansky, 1995, Sachs, 2004]).

Generative Algorithms

Computational models built on the principle of self-organization have their roots in early models of branching structures proposed by Ulam [1962] and Cohen [1967], and within computer graphics in methods that captured the role of light [Greene, 1989, Chiba et al., 1994] and space [Rodkaew et al., 2003, Runions et al., 2007] in shaping the tree. In the model proposed by Takenaka [1994] and brought to computer graphics by Měch and Prusinkiewicz [1996], 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. [2009] extended this idea with a model in which the internal flow of signals also biases the outcome of competition between buds for light. Key parameters of the model controlling the competitive bias and shedding capture tree characteristics that are recognized as visually important by arbourists, 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 [1996], who used a paint program to define the distribution of water in the soil that guided the development of a branching root structure. Rodkaew et al. [2003] and Runions et al. [2007] observed that in self-organizing trees, tree silhouettes can easily be specified by constraining the space within which the trees grow. Neubert et al. [2007] extended Rodkaew's algorithm to incorporate branches that approximate a given input obtained from photographs. Likewise, Côté et al. [2009] adapted the space colonization algorithm of Runions et al. [2007] to proliferate branches around main limbs obtained from LIDAR data. Beneš et al. [2011] 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 modeller 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 Shukri [2007] and Pałubicki et al. [2009] proposed a procedural brush to indicate regions in which the tree grows. Due to the fast response of the underlying tree-generating algorithms, the modeller has the impression of interactively guiding tree growth.

3.2 Procedural Modelling of Landscapes

While much research has addressed the rendering of large-scale landscapes [Deussen et al., 2002, Dietrich et al., 2005], relatively little research has been devoted to the modelling of landscapes since the seminal work of Deussen et al. [1998]. In their system, competition between plants for resources is simulated in two-dimensions to generate a distribution of plant positions. Each plant is associated with a circle defining its neighborhood of influence. As plants grow, if two circles intersect, one plant is eliminated through the application of biologically motivated rules. This process of competition was extended by Alsweis and Deussen [2005] and Alsweis [2007] who allowed for growth of trees to be suppressed by larger neighbors instead of simply eliminating them. The distribution of plant positions is used to control the placement of pre-generated models at render time [Smelik et al., 2011, Beneš et al., 2011]. However, since the plants are generated independently, they lack important characteristics of real world landscapes where plants grow together. The system proposed by Pirk et al. [2012b] addresses this issue by adapting static tree geometry, through pruning and bending intersecting branches, such that neighboring trees fit together. The resulting trees, however, have the appearance of being pruned and bent instead of having been grown together. Moreover, their system only adapts existing trees and therefore a large library of tree models is required.

Instead of generating plant models independently, and later combining them into a scene, growth of a population of plants can be simulated simultaneously. Growth models based on competition are particularity well suited for this as plants can compete for shared resources. Simulation from this perspective dates back to the seminal work of Greene [1989] and Takenaka [1994] and has also been utilized in the more recent work of Měch

and Prusinkiewicz [1996], Rodkaew et al. [2003], Beneš et al. [2009] and Pałubicki et al. [2009]. In these models, neighboring plants compete with each other for space or light, resulting in non-uniform and adaptive tree shapes. However, simulating growth of an entire landscape simultaneously is computationally expensive and, since the trees are grown together, changes to an individual tree requires re-computing the entire landscape.

Multi-level techniques have proven useful in generating large complex landscapes [Deussen et al., 1998]. Lane and Prusinkiewicz [2002] propose a system in which a distribution of tree bases is computed by simulating interactions between tree species such as clustering or inhibition. This distribution defines the placement of predefined crown shapes which are substituted for tree geometry when the tree is rendered. These systems, however, do not simulate the adaptive crown shape of trees growing in the presence of neighbors.

Chapter 4

Generative Algorithms

A *generative algorithm*, or growth model, simulates the growth of a tree by advancing its developmental stage. Generative algorithms determine three key factors: which buds will grow and when (bud fate), the direction of growth, and how long each shoot will be. Interactive procedural modelling of trees requires a generative algorithm that is fast, controllable, and produces a wide diversity of realistic or artistically inspired tree structures. To achieve these objectives, the algorithm must also respect the biological constraints of tree growth. Branch density is one such constraint that many recursive or hierarchical approaches struggle to maintain. Algorithms based on the concept of self-organization inherently handle branch density and have been shown to produce a wide range of realistic tree structures [Pałubicki et al., 2009]. Thus, the generative algorithm presented in this chapter builds on previous self-organizing models. After a brief overview of tree architecture, I will review these previous models in detail and present a new unified model that makes it possible to interactively create a diverse set of natural and artistically expressive trees.

4.1 Tree Architecture

Trees in this thesis are represented as a hierarchy of metamers. Branches are composed of fixed-length linear elements, internodes, characterized by their length, diameter, orientation and position at the proximal end. The orientations are specified using two moving reference

frames (Figure 4.1). The HLU frame is the parallel transport frame [Hanson, 1998], which is closely related to turtle geometry [Prusinkiewicz et al., 2001]. The heading vector H indicates the direction of the internode while the left L and up $U = H \times L$ vectors are used to orient cross sections of the branch geometry. The second reference frame, HVP, is defined by the heading vector H; vector V that is perpendicular to H and lies in the same vertical plane, and vector $P = V \times H$, which lies in a horizontal plane (mnemonically, "parallel to the ground"). This frame is used when calculating tropic and gravimorphic responses of the tree to gravity.

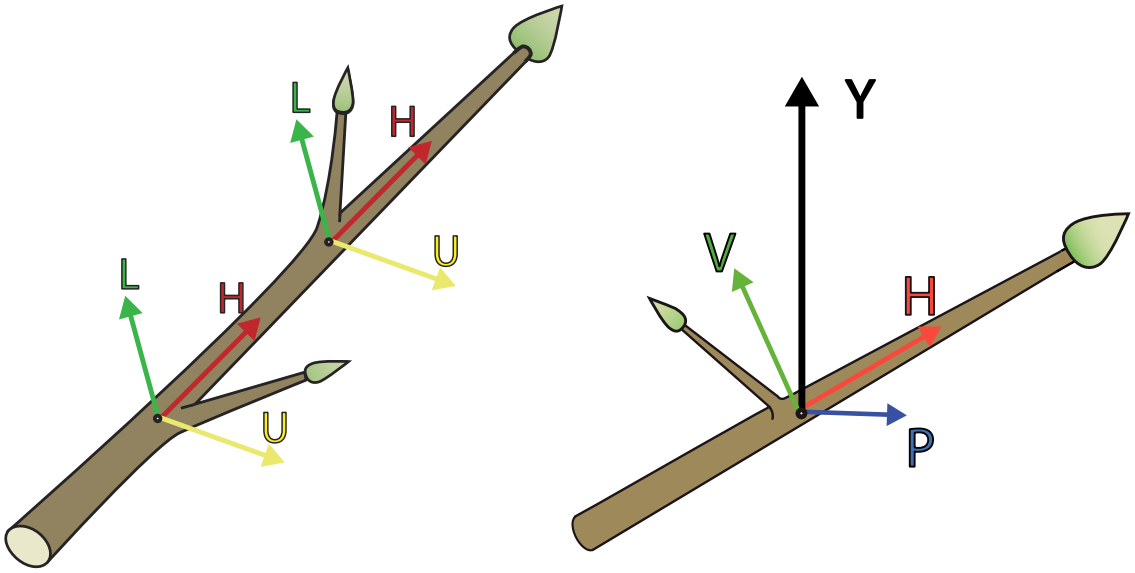


Figure 4.1: Reference frames associated with internodes. From left to right: The ***H***eading – ***L***eft – ***U***p frame and ***H***eading – most ***V***ertical – ***P***arallel to the ground frame. The $\boldsymbol{Y}$ axis represents the global up direction.

A terminal bud is located at the tip of each branch. Lateral buds are positioned at the base of each internode, arraigned in a phyllotactic pattern around their supporting

axis and tilted away from this axis by the branching angle ψ (Figure 4.2). The pattern of lateral bud arrangement, phyllotaxis, can play an important role in the local structure of branching points. A single bud is commonly created at the base of each internode as shown in Figure 4.2. In this case, the buds either form a *distichous* pattern, alternating on either side of their supporting axis, or a *spiral* pattern where a phyllotactic angle, ϕ, defines the rotation between consecutive buds around the axis. Placing multiple lateral buds can increase the diversity of results. In the case of a *decussate* pattern, buds appear in pairs of two on opposite sides of the axis. *Whorls* of buds are also possible where three or more buds are present. When a lateral bud grows it creates a new branch and becomes the terminal bud for this branch. New internodes are created in the path of a growing bud. A shoot refers to internodes created by a growing bud during a single season.

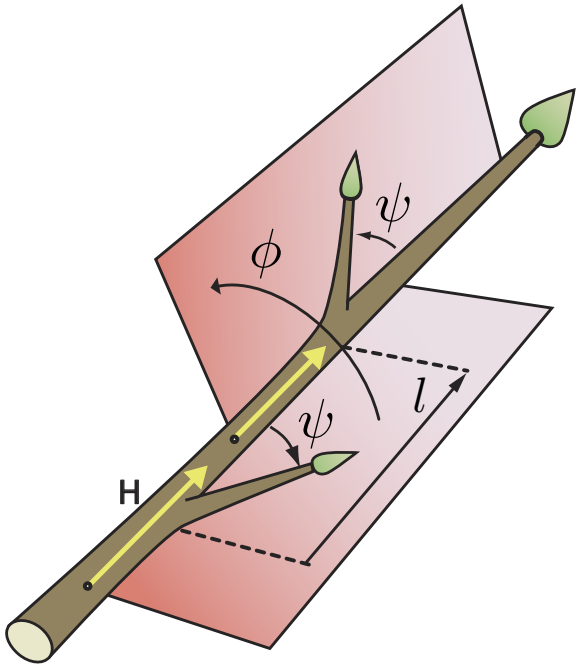


Figure 4.2: Distribution of buds and lateral branches is defined by the internode length l, phyllotactic angle ϕ and branching angle ψ. Figure from [Longay et al., 2012].

4.2 Previous Generative Algorithms

Algorithms based on self-organization generate tree forms by simulating competition between buds for resources. Two classes of self-organizing models have been presented in Pałubicki et al. [2009]. While both models are based on the same core principles, one simulates competition for space while the other for light. The model based on competition for space produces a narrow range of plausible tree structures but allows for interactive guidance of tree development. In contrast, the model based on competition for light has been shown to generate a wider diversity of natural tree forms, but is not directly conducive to interactive control. The unified growth algorithm I present in Section 4.3 builds on these algorithms, therefore I review them both in detail.

4.2.1 Space Colonization

The Space Colonization Algorithm was first proposed as a method for generating leaf venation patterns [Runions et al., 2005] and later extended to create tree structures [Runions et al., 2007]. These initial algorithms lacked architectural constraints such as phyllotaxis and branching angles. These constraints were incorporated by Pałubicki et al. [2009] with the introduction of buds, being the only place on a branch from which a new shoot can grow. The underlying assumption of space colonization is that free space only exists in the vicinity of modeller defined attraction points, markers of free space. All other space is assumed to be occupied. Therefore, placement of attraction points determines where the tree is able to grow, thus controlling the shape of the resulting tree (Figure 4.3).

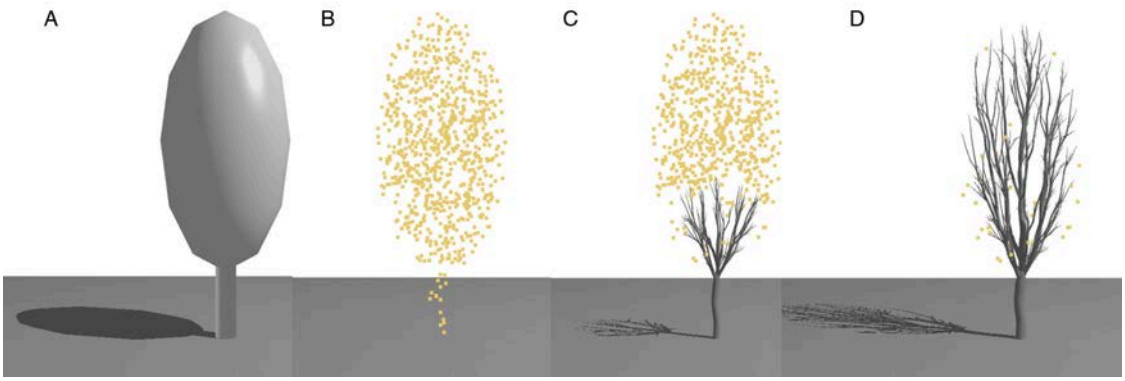


Figure 4.3: Space colonization produces plausible tree forms (generated using the model with buds as presented by [Pałubicki et al., 2009]). A) The desired silhouette is defined. B) Attraction points are scattered within the volume. C) Tree grows towards attraction points (50 iterations). D) All visible points are consumed, points remaining are not visible by any buds (100 iterations).

The iterative operation of the Space Colonization Algorithm is illustrated in Figure 4.5. 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 4.4). At the start of each iteration, attraction points that are within the occupancy zone of one or more buds are removed (e.g., point 1 in Figure 4.4). If any points remain, the buds compete for them according to the following rules:

- An attraction point within the perception volume of a single bud *selects* this bud (points 2 and 3 in Figure 4.4);
- An attraction point within the intersecting volume of two or more buds selects the closest of these buds (point 4 in Figure 4.4);
- Attraction points outside of the volume of perception of any bud are not visible and remain unassociated in a given simulation step (point 5 in Figure 4.4).

Buds are *enabled* for growth if they have at-least one selecting point, all other buds remain dormant. Enabled buds produce a single internode of unit length oriented in the average direction of all selecting attraction points ($\boldsymbol{E}$, Figure 4.4).

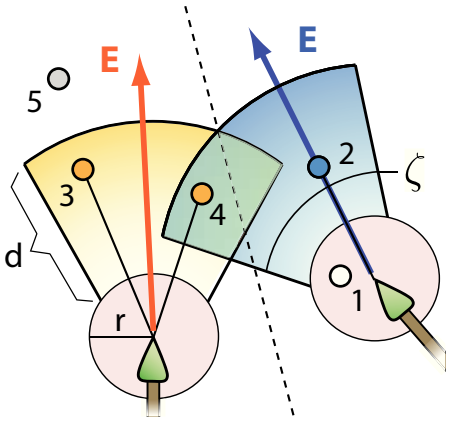


Figure 4.4: Competition of buds for space. Circles of radius r represent the spherical zones occupied by buds. The colored areas represent each bud's volume of perception, characterized by radius $r+d$ and angle ζ. Dots represent attraction points, for which buds compete. The average direction towards the points associated with a bud defines its preferred growth direction $\boldsymbol{E}$. Figure from [Longay et al., 2012]

Attraction points can be added or removed between iterations providing a convenient handle for interaction. Pałubicki et al. [2009] explored this direction by adding attraction points between iterations within a spherical brush controlled by the position of the mouse. This allowed the modeller to guide tree growth into artistic forms, however, the diversity of results was limited. Space Colonization controls the growth of the tree strictly through exogenous environmental signals and its model for bud growth is binary: either a bud grows, or it remains dormant. There is no method to determine which buds should grow more than others. The next section presents an algorithm based on competition for light which incorporates endogenous control resulting in more diverse forms.

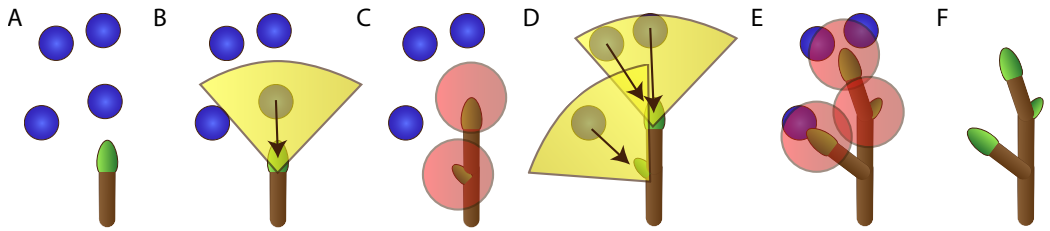


Figure 4.5: Simulating two growth iterations with the Space Colonization Algorithm. A) Initial state, a bud (green) with attraction points (blue). B) Angle of perception and view distance of the bud (yellow). C) The bud creates a new internode, and a new lateral bud is created at the base of this internode. Points within the occupation zones (red) of any bud are removed. D) Two buds compete for space. E) Buds grow in the average direction of their selecting points. F) Final structure.

4.2.2 Competition for Light

Generating tree form by simulating competition between buds for light dates back to the seminal work of Greene [1989]. The growth rate and direction of each bud can be determined by its local light environment. This information can be propagated within the tree to determine whether or not a branch should be shed [Takenaka, 1994, Měch and Prusinkiewicz, 1996]. Borchert and Honda [1984] presented a model of apical control by biasing the flux of resources, flowing from the roots, between main and lateral axes. Pałubicki et al. [2009] incorporated this biasing of resources with competition for light, allowing for the control of forms on the scale from excurrent to decurrent (i.e., with and without a conspicuous trunk).

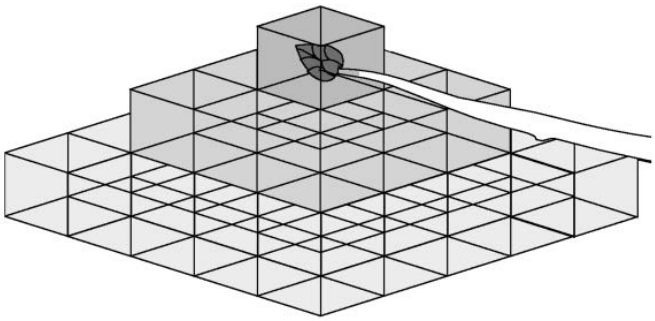


Figure 4.6: In the Shadow Propagation Algorithm, branches cast a pyramidal shadow into a 3D voxel space. Image from [Pałubicki et al., 2009]

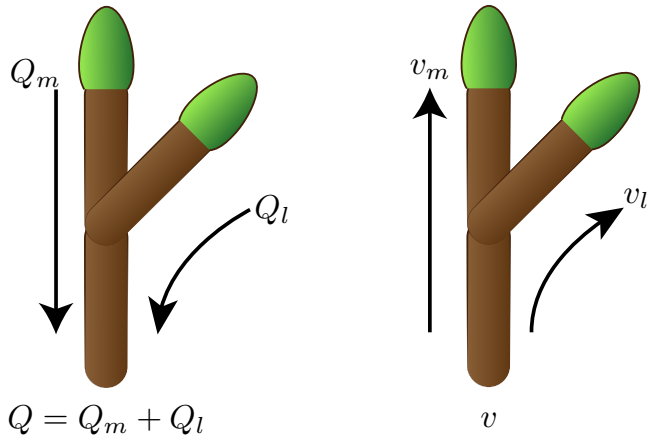


Figure 4.7: Left) Information about the light exposure of each bud is accumulated and stored at each internode as it flows down the tree. For buds $Q = L$. Right) Vigor is then redistributed up from the root as per Equation 4.1. This figure is recreated from Figure 8 of [Pałubicki et al., 2009]

Pałubicki et al. [2009] proposes the Shadow Propagation Algorithm to compute the local light exposure of each bud, L. A voxel space is used to store the light exposure for each point in the scene. Each branch casts a pyramidal shadow into the voxels below it (Figure 4.6). Information about light exposure is sensed at each bud and propagated down

the tree (Figure 4.7). This information is used to determine the allocation of an upwards flowing resource, vigor, which controls how much each bud will grow. Apical dominance is simulated by biasing the flow of vigor at branching points towards the lateral or main axes according to the equation:

$$v_m = v * \frac{\lambda Q_m}{\lambda Q_m + (1-\lambda) Q_l}, \qquad v_l = v * \frac{(1-\lambda) Q_l}{\lambda Q_m + (1-\lambda) Q_l} \tag{4.1}$$

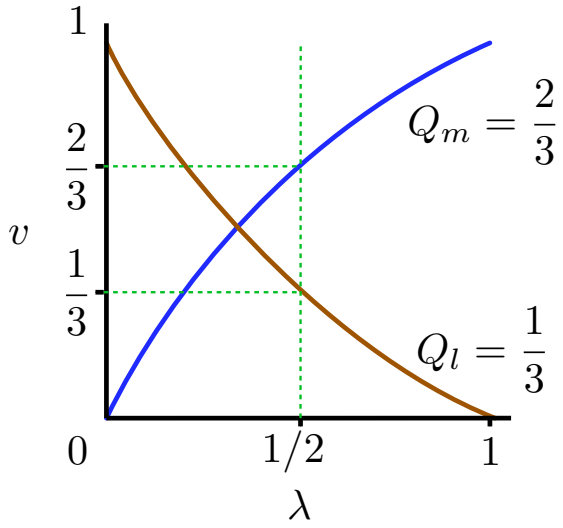


Figure 4.8: Graph of Equation 4.1. With $\lambda = \frac{1}{2}$ the vigor of both lateral and main branches is proportional to their light exposure, Q. Increasing λ allocates more vigor to the main branch at the expense of the lateral. Decreasing λ provides more vigor to the lateral branch at the expense of the main branch.

This equation is illustrated in Figure 4.8. Branches that do not receive sufficient light exposure are deemed a liability for the tree and are shed. The direction of shoot growth is computed using the gradient of light in the shadow voxel space, directing new shoots towards the most illuminated regions. In contrast to space colonization, buds can create

more than one internode per season and buds which receive more resources will generate proportionally longer shoots.

While this algorithm produces a wide diversity of realistic tree forms (Figure 4.9), it is not well suited for interactive control. The growth of the tree could be constrained within the region of a brush, however, brushing downwards from existing branches would not be possible as their shadow would inhibit the activation of new buds. Moreover, branch shedding is used to control the density of branches within the crown. Branch shedding is unintuitive in an interactive setting, as branches that are explicitly created by the modeller can be shed in further stages of development.

Figure 4.9: A winter scene created by trees competing for light using the algorithm presented in [Pałubicki et al., 2009]. Diversity of trees was obtained by editing the source code and parameters of the growth model. Creating even slightly different trees was difficult due to the lack of interactive control.

4.3 Unified Generative Algorithm

This section presents a growth model introduced by Longay et al. [2012] in which development depends *simultaneously* on the availability of light and space. This model combines the potential for high realism with interactive control of tree form. This algorithm is illustrated in Figure 4.10. Beginning with a seedling, branches develop from buds distributed along their supporting axes. Whether a bud will develop into a shoot, and how

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 and terminal vs. lateral buds yielding different tree architectures.

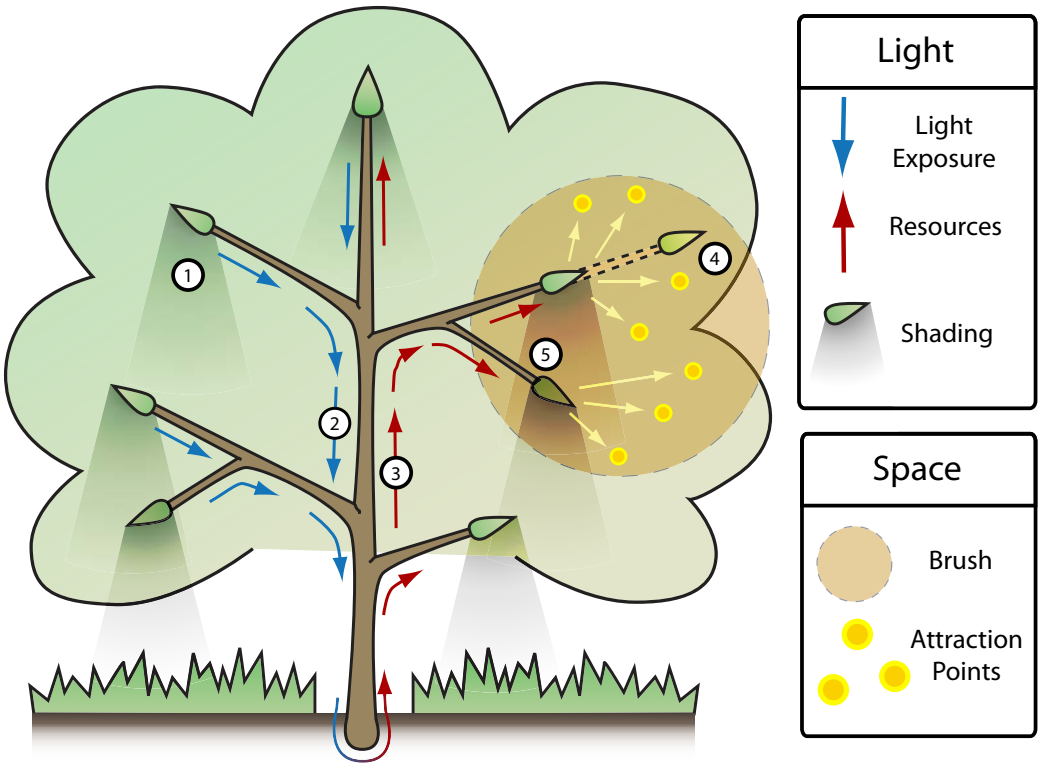


Figure 4.10: A conceptual model of tree development implemented in TreeSketch. 1) Higher-positioned branches cast shadows on lower ones (here shadows are shown only for buds). 2) Information about the light exposure of each bud propagates down towards the tree base (blue arrows). 3) This guides the distribution of a growth-inducing vigor signal flowing back to the buds (red arrows). 4) A bud that has sufficient vigor and wins the competition for attraction points (yellow dots) produces a new shoot. 5) Buds without sufficient vigor remain dormant even if they have space to grow. Figure from Longay et al. [2012]

4.3.1 Competition for Space

Competition of buds for space is modelled as in Section 4.2.1. For directed growth, the set of attraction points can be interactively augmented with new points placed by the modeller. Chapter 5 discusses a brush and sketch-based interface for specifying new points. Autonomous growth (without modeller's guidance) is effected by generating new points within the perception volume of each bud at the beginning of each iteration. All buds produce at least one new attraction point while vigorous buds produce proportionately more, ensuring they will receive space to grow. When a tree grows autonomously, the set of attraction points is cleared at the end of each iteration. This reduces the accumulation of unused points and increases the speed of the algorithm.

4.3.2 Shoot Growth

Superimposed on the competition for space is a process evaluating the vigor of buds. This process determines which of the enabled buds will actually produce new shoots, and how long these shoots will be. This process is modelled using the resource allocation method described in Section 4.2.2, with the following modifications to branch shedding and resource scaling.

Branch Shedding

In nature, different tree species rely on branch shedding to various extents. In an interactive setting, shedding may dispose of a branch that is important to the designer. Therefore, it is preferable to avoid creating superfluous branches rather than shedding them.

Most buds, especially those on the periphery of the tree crown, reside in relatively similar light conditions. Predicting beforehand which of these buds will produce viable

branches is difficult. Therefore, the Shadow Propagation Algorithm activates a surplus amount of buds in each growth phase. Then, to control the density of the tree crown, the algorithm relies heavily on shedding the resulting unproductive branches. A branch is shed if the ratio of its vigor to the size of the branch (measured in number of internodes) falls below a modeller-defined threshold Th [Takenaka, 1994, Měch and Prusinkiewicz, 1996, Pałubicki et al., 2009]. Since this threshold must be set relatively high, it is common to see large branches shed by the tree.

Instead of growing surplus shoots and relying on branch shedding, the unified algorithm activates significantly fewer buds. For a bud to grow it must not only have sufficient resources, but it must also win the competition for available space (Figure 4.10). This considerably reduces the set of potential buds as density is inherently controlled by the competition for space. To decide which of the potential buds to activate, I amplify differences between them, introducing a non-linear response to light exposure. Instead of using the light exposure L of buds directly as the input Q as shown in Figure 4.7, the following relation is used:

$$Q = L^{\kappa} \qquad \kappa \geq 0 \tag{4.2}$$

Parameter κ, set by the modeller, controls the sensitivity of buds to light. When $\kappa = 0$, the growth depends entirely on competition for space. With $\kappa > 1$, fewer buds are activated and the shedding threshold Th can be set considerably lower resulting in a much smaller number of branches being shed. Great results can even be generated without any branch shedding.

Relative Resource Scaling

Branches cast shadows in a downward direction which can inhibit the activation of buds when the modeller attempts to grow downwards from an existing branch. Pałubicki et al. [2009] relies on a constant global scaling factor applied to the value of vigor at the base of the tree to ensure that the tree has sufficient vigor to grow. If this value is not properly set, the tree will either have excessively long shoots, shed too many branches or not grow at all. As presented by Pałubicki et al. [2009], the amount of resource v reaching a bud determines its shoot length n, according to the formula $n = \lfloor v \rfloor$. I propose a more robust and controllable model by computing a vigor scale factor in each step to obtain a desired shoot length.

$$n = \lfloor \frac{v}{v_{max}} n_{max} \rfloor \tag{4.3}$$

Here v_{max} is the highest vigor of any enabled bud and n_{max} is a parameter, set by the modeller, that determines the maximum shoot length. Increased values of parameter n_{max} result in wispy branches, such as those often found in pendulous trees (i.e. birch). When using the brush based interface introduced in Chapter 5, it is possible that only a small set of buds are enabled, all of which may be in shadow. This formula scales the vigor of buds to ensure reliable growth under all conditions.

4.3.3 Parameter Space Exploration

The following figures show results of changing parameters of this unified growth model. All trees presented in this section are grown using the model of autonomous growth. The trees shown in this section are not the most visually appealing but they represent the results of the algorithm in its raw state. In the next section, I introduce modifications that improve

the visual quality and further diversify tree form. The effects of shoot length (parameter n_{max} from Equation 4.3) is best shown in conjunction with a downwards tropism to create long pendulous branches, therefore, it is presented in the next section.

Figure 4.11: The density of the branching structure is changed by modifying the radius r from Figure 4.4. Left to right, $r = 1, 15, 30$. Units are measured in internode lengths (Figure 4.2)

Figure 4.12: The effects of non-linear response to light exposure on tree form (parameter κ from Equation 4.2). Left) $\kappa = 0.5$, surplus buds are activated, all buds grow into roughly uniform length shoots. Right) $\kappa = 2.5$, only buds in good light conditions are provided resources to grow resulting in a similar but less dense structure. In both cases no branch shedding is used. All other parameters remain constant.

Figure 4.13: The effects of apical dominance on tree form. Parameter λ from Equation 4.1 is changed from left to right (0.46,0.48,0.5,0.52,0.54). Trees are grown to roughly the same density with autonomous growth. All other parameters remain constant.

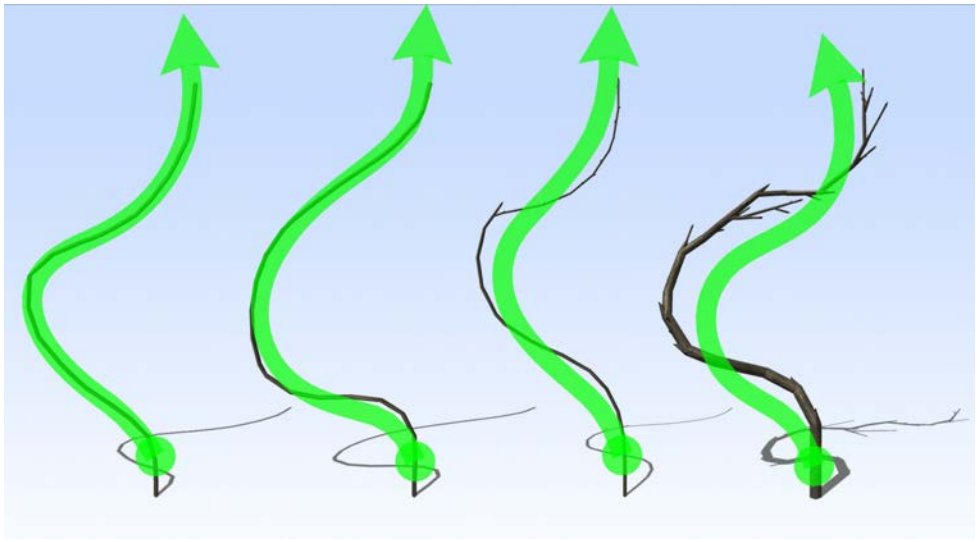


Figure 4.14: In each case, a dense set of attraction points was created along the path of the green arrow. The parameter controlling branch straightness α changes from left to right (0, 0.3, 0.7, 1.0). All other parameters remain constant (except for γ, which is adjusted appropriately). Increasing straightness reduces interpolation of the points and in the extreme case, changing direction relies on branching.

4.4 Diversifying Form

This section presents additional features which extend the diversity of tree forms while broadening the biological foundation of the algorithm.

4.4.1 Branch Straightness

In the basic growth model, terminal buds grow in a direction controlled by the environment. This leads to branches that suddenly change course and can look unnatural. As in [Pałubicki et al., 2009], the growth direction, $\boldsymbol{H}'$, is computed by blending the direction from the environment $\boldsymbol{E}$ (Figure 4.4), with the current heading of the bud $\boldsymbol{H}$, as described by the in equation:

$$\boldsymbol{H}' = \alpha\boldsymbol{H} + \gamma\boldsymbol{E}, \quad \alpha + \gamma = 1 \tag{4.4}$$

The parameter α has the effect of increasing the straightness of branches. Figure 4.14 shows examples of this parameter on a single branch, and Figure 4.15 shows the result across a whole tree.

Figure 4.15: The parameter controlling branch straightness α changes from left to right, (0, 0.5, 1.0). All other parameters remain constant (except for γ, which is adjusted appropriately). With high branch straightness the branching angle of the tree greatly influences its form. Trees generated by autonomous growth.

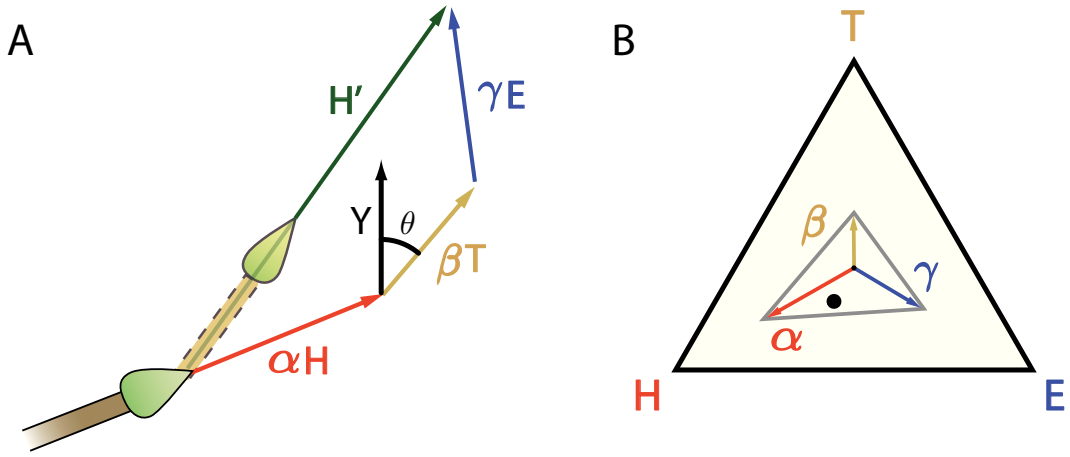


Figure 4.16: A) New growth direction $\boldsymbol{H}'$ is a weighted sum of the current bud direction $\boldsymbol{H}$, the gravitropism vector $\boldsymbol{T}$, and the preferred growth direction $\boldsymbol{E}$ (Figure 4.4). B) Visualization of weights α, β, γ as a point in the triangle $\boldsymbol{HTE}$ (black dot), and as vectors from the gravity center of this triangle to its vertices (arrows). Figure from Longay et al. [2012]

4.4.2 Gravitropism

Chapter 2 discussed the importance of gravitropism on tree form. Pałubicki et al. [2009] modeled gravitropism by adding a weighted vector in the desired direction to the growth direction of each bud. However, this does not allow for the radially symmetric tropisms such as plagiotropism (the tendency to maintain a slanted or horizontal orientation). To incorporate gravitropism into the growth model, the direction of tropism is defined by a modeller-specified angle θ. Specifically, for each bud a local tropism vector $\boldsymbol{T}$ is calculated by rotating the global up vector $\boldsymbol{Y}$ by θ in the vertical plane which includes the current direction of the bud $\boldsymbol{H}$. If $\boldsymbol{H}$ is almost vertical, symmetry is broken by choosing the vertical plane of rotation at random. As shown in Figure 4.16A, the new growth direction, $\boldsymbol{H}'$, is then calculated as the weighted sum (extending Equation 4.4):

$$\boldsymbol{H}' = \alpha\boldsymbol{H} + \beta\boldsymbol{T} + \gamma\boldsymbol{E} \tag{4.5}$$

$$\alpha + \beta + \gamma = 1$$

$$0 \leq \alpha, \beta, \gamma \leq 1$$

where $\boldsymbol{E}$ is the unit vector corresponding to the optimal direction of local growth obtained from the competition for attraction points (Figure 4.4). Parameters α, β and γ control the relative influence of current bud direction (straightness), 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 4.16B). This visualization is used in the design of the widget for manipulating these parameters interactively (Chapter 5) The effects of gravitropism can be seen in Figures 4.17 and 4.18.

Figure 4.17: The effect of plagiotropism on tree form. Tree is initially grown upwards then grown outwards with strong plagiotropism (horizontal).

Figure 4.18: The effects of shoot length on tree form (parameter n_{max} from Equation 4.3). Tree is initially grown upwards then a strong downwards tropism is applied. Left) $n_{max} = 2$, causes short shoots. Right) $n_{max} = 6$, longer shoots results in drooping pendulous branches.

In many trees, different branches exhibit different gravitropic behavior [Hallé et al., 1978]. For example, in a conifer tree, the main axis grows upwards while lateral branches tend to grow horizontally. To model such trees, the tropism of the main axis can be optionally constrained upwards, different from all other branches. Figure 4.19 shows a conifer tree generated using this method.

Figure 4.19: A conifer tree created by setting the tropism of the main axis upwards while all other branches are plagiotropic. Tree is generated by autonomous growth.

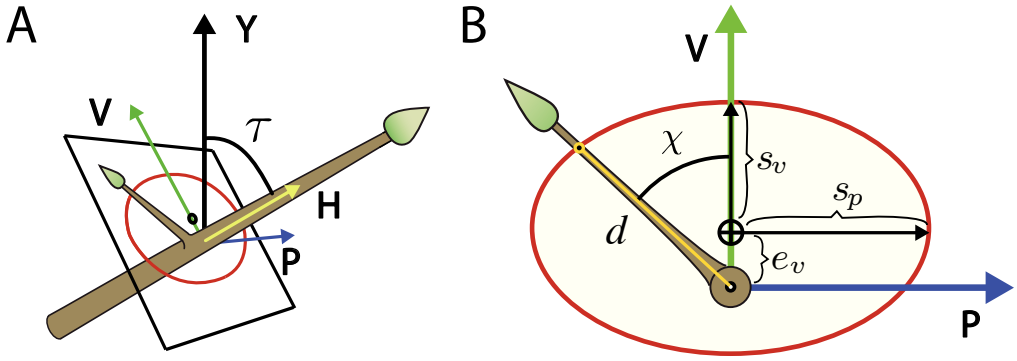


Figure 4.20: Specification of the gravimorphic response. The axes s_v and s_p of the control ellipse are aligned with axes $\boldsymbol{V}$ and $\boldsymbol{P}$ of the $\boldsymbol{HVP}$ ($\boldsymbol{H}$eading – most $\boldsymbol{V}$ertical – $\boldsymbol{P}$arallel to the ground) reference system associated 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.

4.4.3 Gravimorphism

As discussed in Chapter 2, gravimorphism refers to the differential growth potential of buds that depends on the orientation of the bud with respect to its parent axis. To quantify this phenomenon, an ellipse is defined lying in a plane perpendicular to the parent axis at the location of a bud (Figure 4.20A). The propensity b of this bud to grow (characterized further down) 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 4.20B). Since gravimorphism is most predominant on horizontal branches, its strength is defined depending on the angle τ between the axis direction $\boldsymbol{H}$ and the vertical direction $\boldsymbol{Y}$. Grouping these factors together, the effect of gravimorphism is captured by the equation:

$$b = \cos^2 \tau + d \sin^2 \tau \tag{4.6}$$

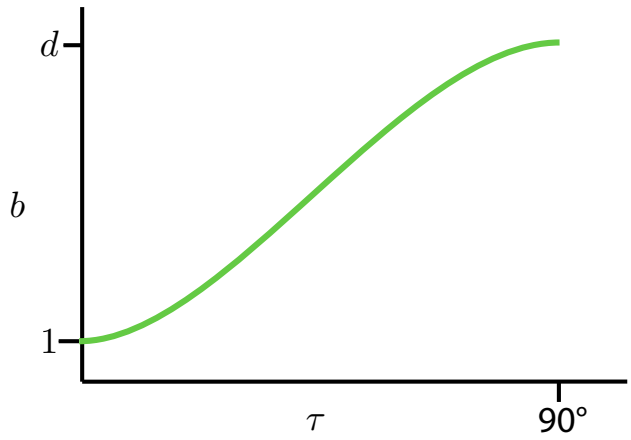


Figure 4.21: Graph of Equation 4.6.

A graph of Equation 4.6 is shown in Figure 4.21.The strength of gravimorphism, b, is incorporated by modifying how buds report light exposure (Section 4.2.2). This is accomplished by introducing into Equation 4.2, the strength of gravimorphism b as follows:

$$Q = bL^{\kappa} \tag{4.7}$$

This captures the differential development of lateral buds depending on their orientation on horizontal or slanted parent axes. Gravimorphic responses of different types and magnitudes can be defined by manipulating the position and displacement of the control ellipse (Section 5). Gravimorphism and gravitropism often work in concert, producing dramatically different tree forms. Examples are shown in Figure 4.22.

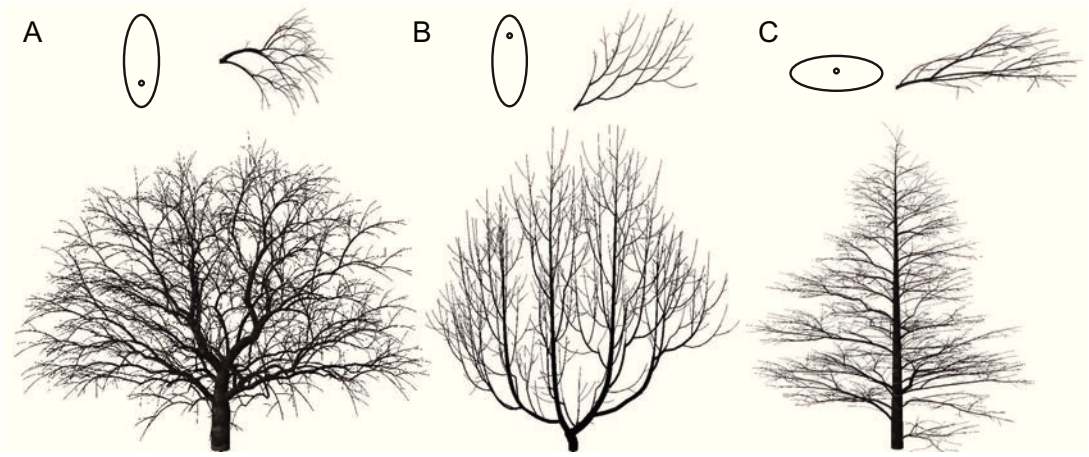


Figure 4.22: Contrasting tree forms resulting from the interplay between gravitropism and gravimorphism. Trees generated by autonomous growth. A) Downward tropism and epitony. B) Upward tropism and hypotony. C) Plagiotropism and amphitony.

4.4.4 Deflection Angle

The growth of a lateral bud can divert the path of its supporting branch as shown by Prusinkiewicz et al. [1997]. This effect is captured by the deflection angle δ (Figure 4.23). When the value of δ is equal to that of the branching angle ψ, junctions form a characteristic Y shape which can be seen in many trees. With small angles, this parameter can be used to give tree structures a gnarled character.

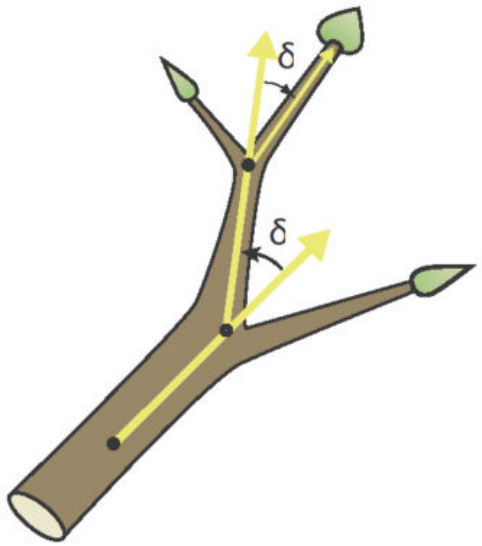


Figure 4.23: At the branching point, the main branch may deflect in the direction opposite to the lateral bud by angle δ. Figure courtesy of Adam Runions.

Figure 4.24: The effect of the deflection angle on tree form. The angle δ changes is 0° for the tree on the left and 30° for the tree on the right. All other parameters remain constant. Trees generated by autonomous growth.

4.5 Discussion

In this chapter I presented a new growth model which retained the high level of realism obtained by previous models while allowing for interactive control. I have increased the diversity of previous models through the incorporation of gravimorphism and tropisms quantified using the gravitropic set-point angle. The next chapter introduces interactive widgets to control parameters of growth and a brush-based multi-touch interface for creating trees. Further results utilizing this growth model are presented throughout the remainder of this thesis, in particular Sections 5.6 and 6.9.

Chapter 5

TreeSketch

This chapter presents TreeSketch, a program that allows the modeller to create a wide range of trees by combining procedural methods with detailed control of tree form. The system integrates the procedural growth model discussed in Chapter 4 with a multi-touch tablet interface that provides detailed control of tree form. The modeller can control the tree structure 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. TreeSketch is on its third public release for the Apple iPad and has obtained over fifty-thousand downloads.

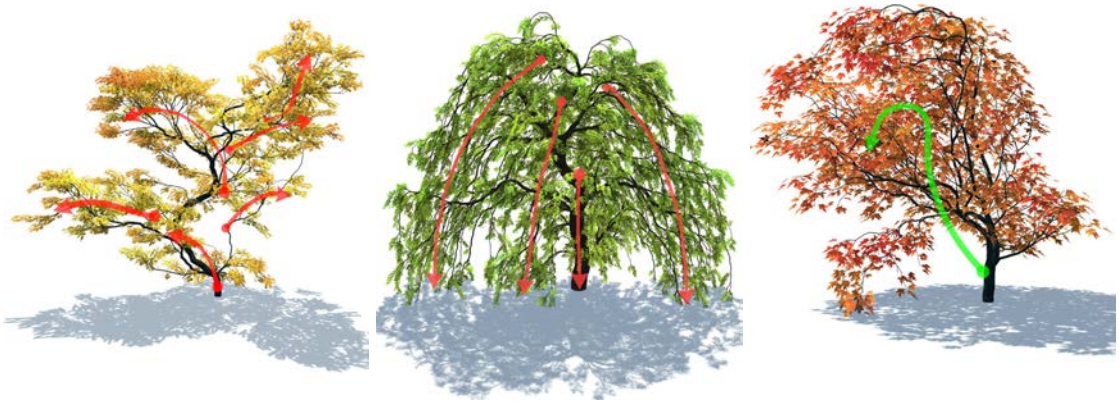


Figure 5.1: Examples of trees created with TreeSketch. Arrows indicate the motions of the brush that determined the corresponding tree forms.

The main tool is a brush, which gives the modeller decisive control over the form of the tree (Figure 5.1). By changing the brush radius, the modeller can seamlessly progress from specifying the exact course of individual axes to a broad definition of large branches and the entire crown. The modeller 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 5.2 shows an example of editing the tree structure). Branch widths are defined procedurally, but can be modified 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 a wide diversity of natural and artistically expressive trees.

5.1 Interface

An important design goal was to create a seamless interface allowing modellers to freely intermix operations of growing, manipulating and viewing the tree. Growing and manipulating the tree structure are 'first class citizens', being controlled with a single touch, while modifications to the view are invoked through multi-touch gestures. At any stage of the design, the tree can be translated by sliding two touches across the screen and zoomed in or out with the standard two-touch pinch gesture. Sliding three touches across the screen grabs and tilts the tree toward the modeller (with vertical motion) and around the axis of

its trunk (horizontal motion). A rotation strip (Figure 5.3) is provided along the bottom of the screen for rotating around the axis going through the trunk of the tree . While rotation about this axis can also be invoked with three touches, this strip allows for easily rotating the tree *while* painting, allowing for the creation of novel tree forms such as a spiraling trunk (Figure 5.10).

Figure 5.2: Designing a bonsai-inspired tree. The modeller 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 branches with several strokes of a larger brush, increasing the girth of the main trunk, and stretching the trunk vertically (right). Figure from Longay et al. [2012].

Figure 5.3: Labeled overview of the TreeSketch interface.

5.1.1 Manipulating Growth Parameters

The bar located at top of the screen (Figure 5.3) allows the modeller quick access to tabs that control the parameters of the model and rendering system. When these buttons are pressed, the tab drops down and the work area is reduced in size but remains active. Below is a description of the most important tab, the Architecture tab, which presents manipulable diagrams to control procedural aspects of the model. The remaining tabs are shown in Appendix E.

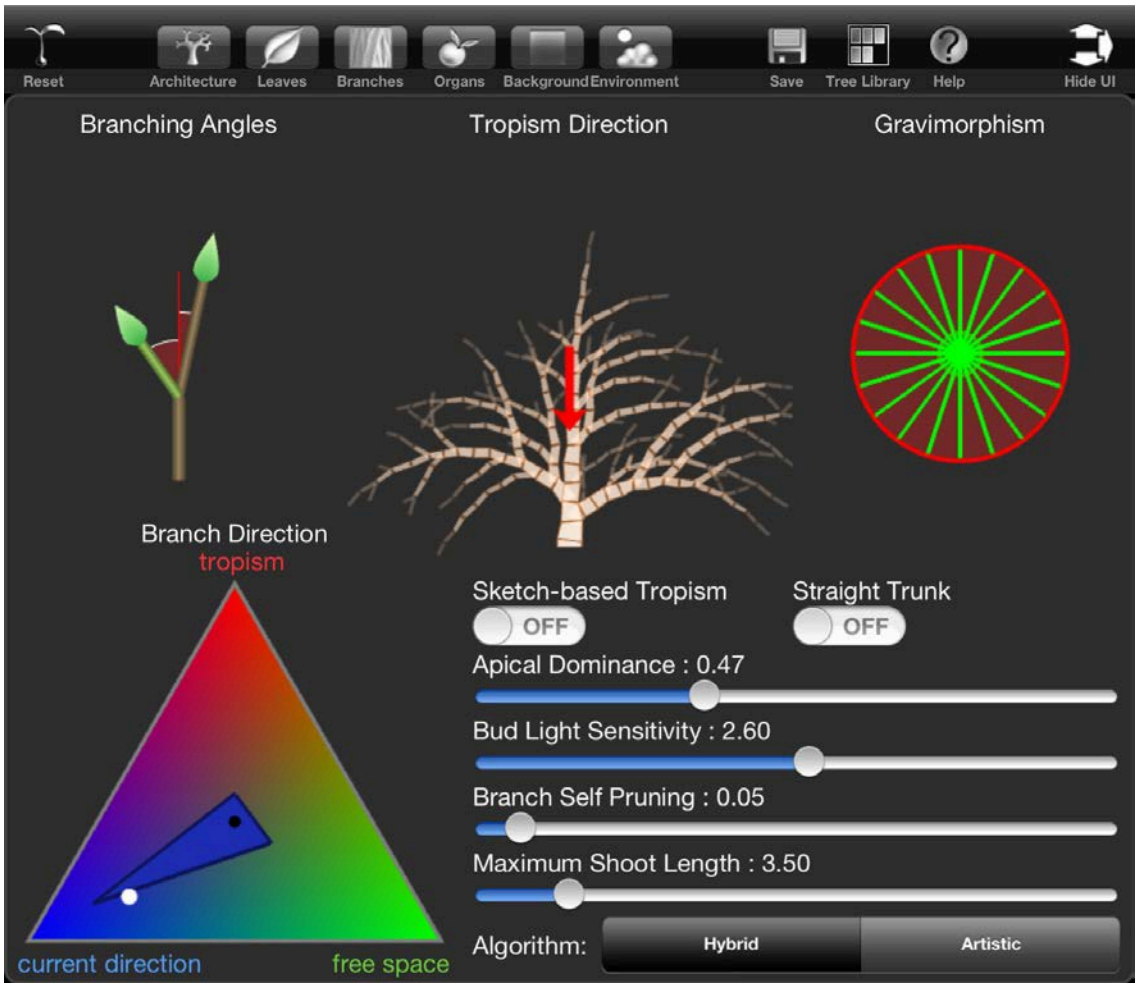


Figure 5.4: The TreeSketch architectural panel.

The architectural panel (Figure 5.4) allows the modeller to control general characteristics of the modeled trees by manipulating key parameters of the generative algorithm. As discussed in Chapter 4, the effects of tropism, gravimorphism and current branch direction are correlated. Therefore, this control panel is designed to bring together these important parameters. The widgets are designed such that several parameters can be edited at once providing a higher level control. A central component of this panel is the two-dimensional schematic tree model behind the tropism direction widget, which is generated using the same growth model as the main tree but constrained to two-dimensions. This model changes form instantaneously as the growth parameters are modified, providing the modeller with real-time visual feedback.

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 4.16A. Sliding a touch across the region 'pushes' the arrow defining tropism direction and modifying its angle with respect to gravity. The strength of tropism is defined by the branch direction widget discussed below.

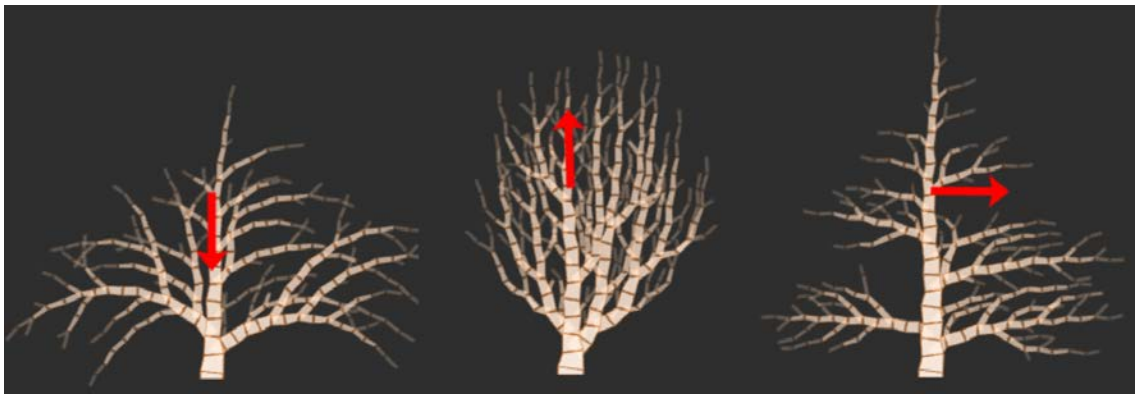


Figure 5.5: Tropism direction widget. Left) Downwards tropism, Center) Upwards tropism, Right) Horizontal tropism with the trunk constrained to be vertical.

Gravimorphism

This widget controls the preferential growth of buds according to their orientation on the supported axes. The gravimorphism widget is a manipulable version of the ellipse in Figure 4.20B, returning values of parameters s_v, s_p and e_v. The lines emanating from the center of the widget represent bud directions, and their color, transitioning from red to green, corresponds to the relative quantity of resources (small to large, respectively) a bud in this orientation will receive. The ellipse is manipulated by sliding it vertically using a single touch or pinching with two touches to scale it in either direction.

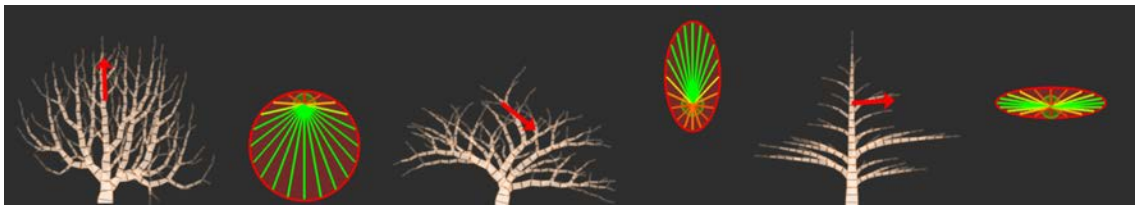


Figure 5.6: Left) Ellipse shifted down. Buds on the underside of their supporting axis are preferred for development (hypotony). Center) Ellipse shifted up and pinched horizontally. Buds strictly on the upper side of their supporting axis will be preferred for development (amphitony). Right) Ellipse pinched vertically. Buds in a horizontal position on their parent axis will be preferred (epitony).

Branching Angles

The branching angle widget allows for interactive control of the branching angle, ψ (Figure 4.2), and the deflection angle, δ (Figure 4.23). Both the main and lateral buds can be rotated to adjust their corresponding angle.

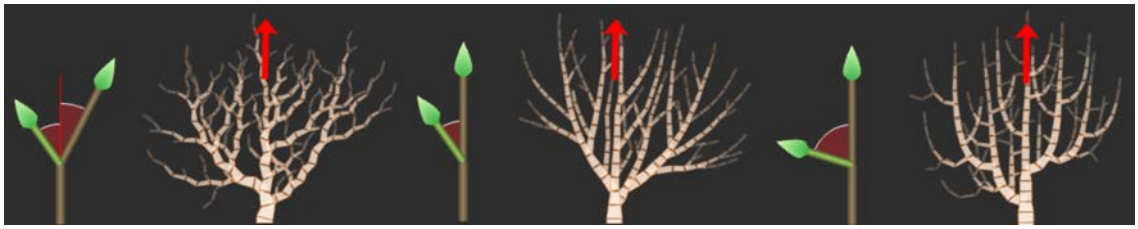


Figure 5.7: Branching angles widget. Left) A large deflection angle results in a gnarled structure. Center) Reducing the deflection angle produces smooth branches. Right) Large branching angles can result in swooping branches.

Branch Direction

The direction of shoot growth is computed as an affine combination of the direction of tropism, current bud direction and a direction from the environment (Figure 4.16). Since their weights are interdependent, it is not appropriate to use sliders to control them because

changing one slider would require changing the value of the other two. The branch direction widget is a manipulable version of Figure 4.16B, returning values of weights α, β and γ. The modeller manipulates the current value by pushing the white dot around the triangle (Figure 5.8). A smaller triangle at the center of the widget is used to visualize the weights by scaling the distance of each vertex from the center depending on its corresponding weight. This idea was inspired by the concept of star glyphs used for visualizing multi-dimensional data points [Ward, 1994].

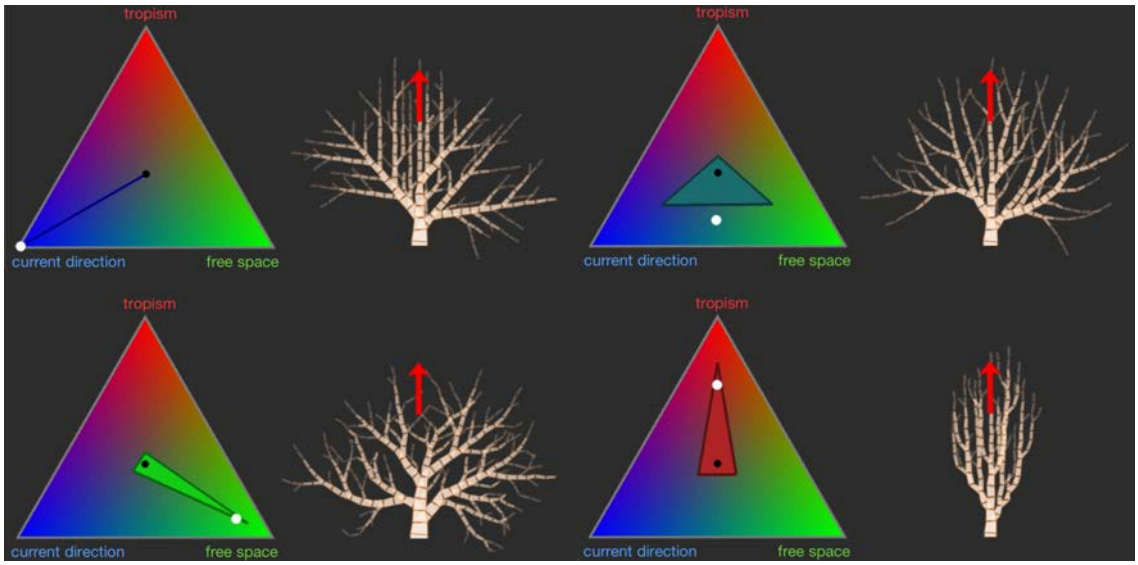


Figure 5.8: Branch direction widget. The position of the white dot within the triangle controls the relative weighting of tropism, current direction and free space (environment).

Remaining Parameters

In addition, apical dominance, λ, 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 (discussed further in Section 5.2.3) and setting an upward tropism of the trunk independently of the

tropism of the branches (useful when modeling conifers). The buttons relating to *Algorithm* are purely for backward compatibility with previous versions of the software and are not discussed in this thesis.

5.1.2 Side Bar

The side bar (Figure 5.3) provides control of important tools used when growing the tree. Hidden sliders emerge when pressing one of the top two buttons, Brush and Grow (Figure 5.9). Hiding these sliders while not in use allows all tools to fit within the limited screen space and maintains the clean, minimalist look of the interface.

The brush slider modifies the radius of the spherical brush used to distribute attraction points that direct the growth of the tree (discussed further in Section 5.2.1). As this slider emerges, the position of the bar shifts vertically to keep the button under the touch, allowing its position to reflect the current brush radius. This ensures all adjustments to the brush radius are relative to its current value. The size of the brush radius (measured in screen space) is displayed on the screen whenever the Brush button is held down. Changing of the brush radius while painting is extremely useful (Figure 5.11). Reflecting this usefulness, the position of the side bar can be switched between the left and right side of the screen to accommodate both left- and right-handed modellers.

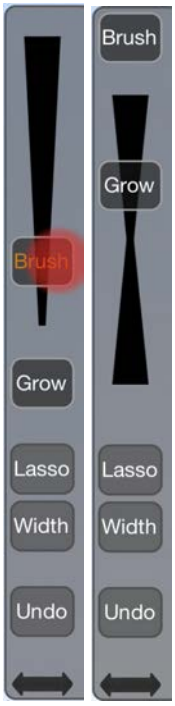


Figure 5.9: The expanded state of the side bar when the Brush or Grow buttons are held down.

A second slider emerges when the Grow button is pressed allowing the modeller to adjust the speed of autonomous growth (pulled upwards) and continuous undo (pulled downwards, 'ungrowing' the tree). When this button is released, growth is halted and the slider disappears. Continuous undo rolls back the state of the tree to an earlier time by small intervals.

The Undo button rolls back the model by entire operations such as a stroke of the brush. The remaining buttons are discussed in the context of their operations in the next section.

5.2 Artistically Driven Tree Growth

The modeller can guide tree growth interactively by distributing attraction points using a procedural brush or by defining or extending the envelope of the tree with a lasso tool.

5.2.1 Brushing and Sketching

The *brush* is implemented by continually generating attraction points (with uniform random distribution) within a sphere invoked and manipulated by the modeller using single-finger strokes (Figure 5.10, 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. As stated previously, the radius of the brush can be changed while brushing (Figure 5.11). As the radius of the brush decreases to zero, brushing transitions to sketching in a continuous manner (Figure 5.10, right).

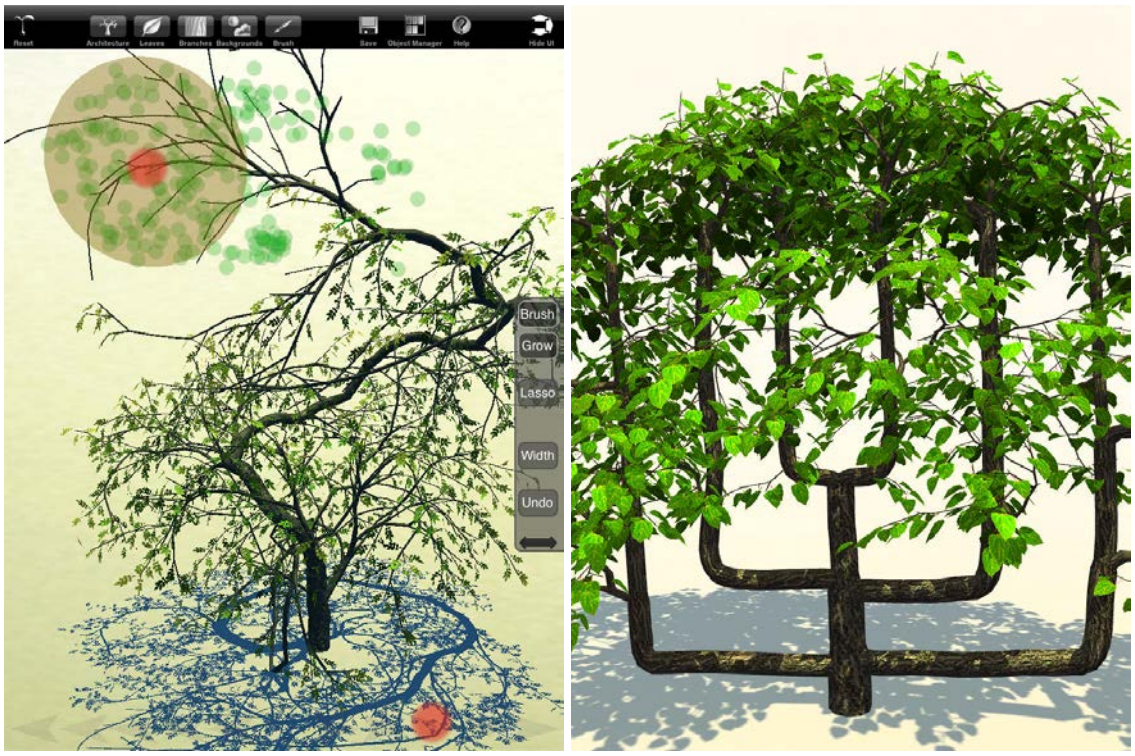


Figure 5.10: Brushing and sketching. Left) A snapshot of the TreeSketch screen. The tree was rotated while brushing, resulting in a helical trunk. Pink dots indicate position of fingers on the tablet. Right) A candelabra espalier created by sketching the main branches, then brushing the upper crown. Figure from Longay et al. [2012].

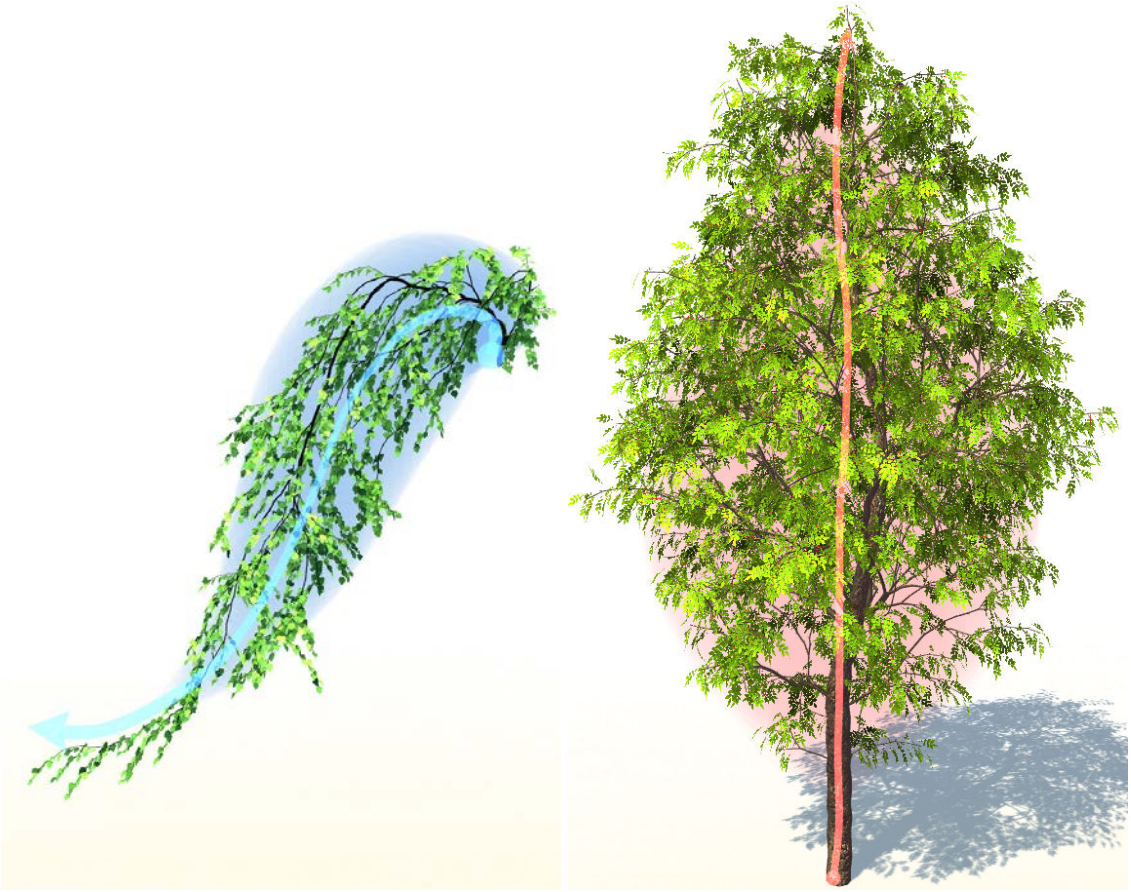


Figure 5.11: A hanging branch and a tree created with single strokes of the brush (arrow) with interactively changed radius (visualized as colored sweeps). Figure from Longay et al. [2012].

5.2.2 Lasso

The *lasso* is used to sketch the silhouette of the entire tree or its part (Figure 5.12). When the 'Lasso' button is held down, single strokes are interpreted as 2D silhouette sketches. The same rules for inferring the depth of the stroke in the case of brushing are used here. To infer a 3D envelope from this 2D sketch, image-based erosion of the silhouette is first used to infer its chordal axes [Hall, 1989]. The silhouette is then inflated around these axes

as in the Teddy system [Igarashi et al., 1999]. The inflated envelope is filled with attraction points by generating random points within a cube bounding the envelope, and retaining those points that fall within the envelope.

Figure 5.12: Examples of tree forms controlled with a lasso (red contours shown for two of the models). Figure from Longay et al. [2012].

5.2.3 Sketch-based Tropisms

When artists paint with a brush, the bristles of the brush leave groves in the paint, indicating the direction of the stroke. This effect can be used, for example, to provide structure to coarsely painted foliage as shown in Figure 5.13.

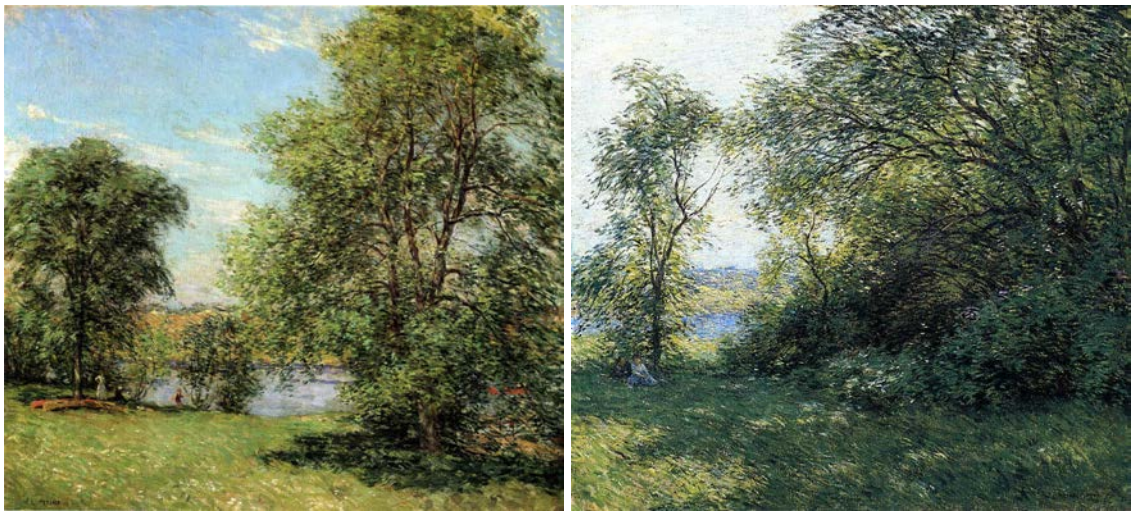


Figure 5.13: Examples of paintings that utilize brush stroke direction to indicate the structure of foliage. Paintings by [Metcalf], Left: The Boat Landing (1902), Right: The Bower (1907).

As discussed in previous chapters, tropism defines the tendency of branches to grow in a particular direction. In previous systems, the strength and direction of this parameter was set as a uniform global value [Měch and Prusinkiewicz, 1996], changed as a function of developmental stage [Pałubicki et al., 2009], or interpolated from a collection of point attractors [Aono and Kunii, 1984] . In TreeSketch, when creating a tree interactively, the direction of tropism can be coupled with the direction of the brush stroke. This provides the modeller with the feeling that they control the flow of branches while brushing, significantly enhancing the expressiveness of brush strokes as shown in Figure 5.1. With this feature enabled, the branches take on the style of brush strokes as shown in Figure 5.14

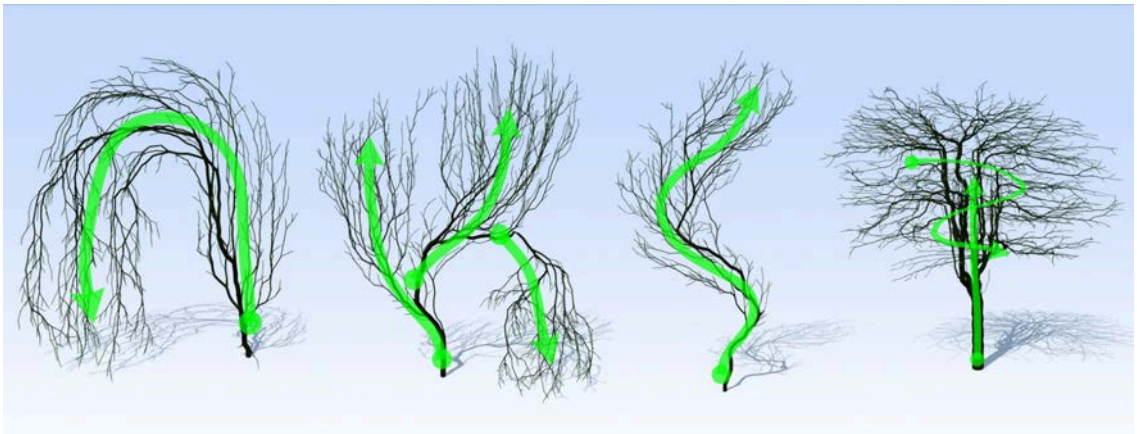


Figure 5.14: Example trees created with sketch-based tropism. Green arrows represent brush movement.

5.2.4 Selective Growth

By default, all buds associated with at least one attraction point are considered enabled: they will produce new shoots if they are 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 modeller. To enter this mode, the modeller presses and holds a point on a branch from which they want to grow. The selection is confirmed by an animation of expanding red circles. Due to the discrete nature of the generated structure, it is possible that the closest bud to the selection point is not oriented in the direction of the brush stroke. To ensure reliable growth, TreeSketch re-orients the nearest bud in the initial direction of the brush stroke. In the selective mode, all buds remove markers within their occupancy zones, but only buds in the new branch may grow after competing for the remaining markers (Figure 5.15).

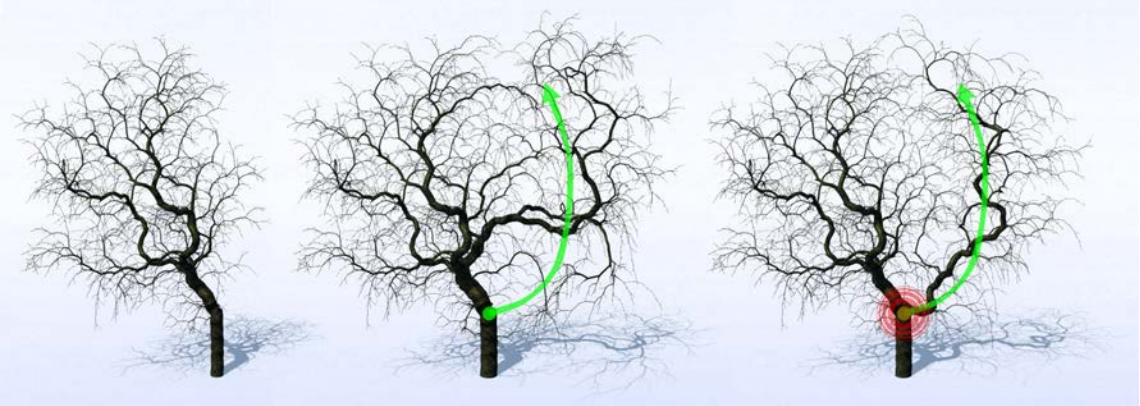


Figure 5.15: Comparison of non-selective and selective growth. The initial tree crown (left) was refined using non-selective growth (middle) and selective growth (right). Non-selective growth produced a series of branches extending from the initial tree lobe, whereas selective growth produced a separate lobe from the selected bud indicated by the expanding red circles. Figure from Longay et al. [2012].

5.3 Manipulating Tree Form

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 three editing operations: bending branches, selective modification of branch width and pruning.

5.3.1 Branch Bending

Branch bending provides a convenient metaphor for editing trees, consistent with the physical manipulations employed by horticulturists when training trees. In computer graphics, bending was introduced as a method for editing plants by Power et al. [1999],

who implemented it using inverse kinematics. After reimplementing and tested their technique, it was noticed that while it produced plausible deformations, it suffered from two limitations. First, it did not support circular editing: moving the end effector back to its original position did not restore the branch to its original form. Second, manipulations did not preserve the local character of branches (Figure 5.16).

Figure 5.16: Comparison of deformation methods. The ghosted branch represents initial state and red arrow shows the desired bending operation. A) Bending with inverse kinematics results in a loss of local features. B) Bending a branch with the PriMo-based method described in the text preserves the local character of the branch. C) Increased prism width favors stretching of joints over bending, resulting in a more rigid deformation.

In the broader context of surface editing, similar limitations were overcome in the PriMo mesh-based surface deformation method [Botsch et al., 2006]. I have adapted PriMo for the purpose of deforming tree skeletons. PriMo represents mesh geometry as a set of rigid prisms connected by elastic joints. The modeller 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. A detailed description of this branch deformation method is presented in Appendix A.

The PriMo method supports circular editing [Lipp et al., 2011] as moving the selected prisms back to their initial location undoes the deformation. As noted by Botsch et al. [2006], scaling the radius of the prism geometry provides control over the susceptibility of joints to stretching and bending (Figure 5.16). Although real tree branches are almost non-stretchable, it is convenient to allow for stretching when editing branches, which is particularly useful to elongate a trunk.

The tree is manipulated by changing the position of the selected internodes with a multi-touch gesture (Figure 5.17). As an axis is deformed, other branches maintain their relative positions and orientations with respect to their supporting internodes. Bending branches can be used to create a variety of novel tree forms as shown in Figure 5.18.

Figure 5.17: 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 internode E, respectively. The in-between touch provides an additional positional constraint. Figure from Longay et al. [2012].

Figure 5.18: Novel tree forms which relied heavily on branch bending in throughout their creation. Figure from Longay et al. [2012].

5.3.2 Width Constraints

The width of tree branches can be predicted by the formula,

$$d^n = d_1^n + d_2^n \tag{5.1}$$

which relates the diameter d of an internode below a branching point to the diameters d_1 and d_2 of the internodes above [MacDonald, 1983]. 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 [1983]

pointed out, however, that other values of this exponent, usually between 2 and 3, may be more realistic. 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 $M+1$ terminal branches with diameters $d_0, ..., d_M$ will thus have diameter d that satisfies the equation:

$$d^n = \sum_{i=0}^{M} d_i^n. \tag{5.2}$$

However, these equations are only suitable for younger tree structures. Old, mature trees tend to have branches and trunks that appear thicker than the formula predicts. This is due to the creation of biomass to support branches that were shed or broken in later stages of development. The degree of freedom provided by parameter n is used to incorporate modeller-defined constraints into the computation of branch widths. In the simplest case, the modeller specifies the width of branches at the extremities (d_e) and at the base of the tree (d_b). From Equation 5.2 it then follows that:

$$n = \frac{\log M}{\log d_b - \log d_e} \tag{5.3}$$

In addition the modeller can introduce any number of constraints at arbitrary locations on the tree (Figure 5.19). These locations partition the tree into subtrees and Equation 5.2 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 5.2 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.

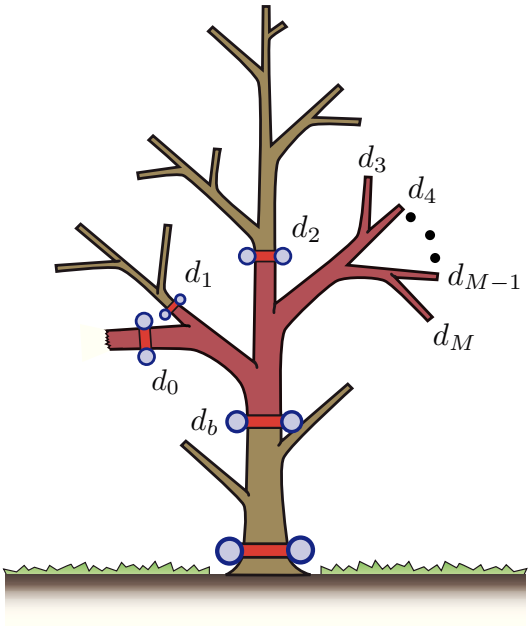


Figure 5.19: Width constraints (red) divide a tree into subtrees. Within each subtree, Equation 5.2 is solved to yield the exponent *n*, then branch width is calculated using the generalized da Vinci formula. Figure from Longay et al. [2012].

A width constraint can be placed on a branch by holding the Width button on the Side Bar while selecting the desired branch. The width is then adjusted by pulling either of the handles in the direction perpendicular to the branch axis. Selective changes in branch width have a significant impact on the appearance of the tree (Figure 5.20).

Figure 5.20: A model (left) is refined by modifying the width of selected branches (right). The model on the right has the character of a more mature tree. Figure from Longay et al. [2012].

5.3.3 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 5.19).

5.4 An Augmented Reality Interface

This section presents an augmented reality interface which allows the modeller to create a tree in the context of a real world environment. This interface not only provides an immersive and fun experience, but also has functional advantages that can improve the

modelling process. Depth throughout brush strokes can be conveniently modified by moving the tablet in and out during the sketch. Large strokes are also possible by panning the tablet while sketching. Furthermore, the environment the tree is modeled in can play an active roll in the tree's development. Trees take a significant amount of time to grow, with the overall shape of the tree changing throughout the development. The ability to grow the tree and visualize future stages of development makes this interface particularly useful in landscaping applications.

5.4.1 Scene Tracking

Augmented reality tracking is performed by the PointCloud SDK [13th Lab AB, 2012]. For each input video frame, the PointCloud library returns a matrix, corresponding to the current camera transformation, along with the set of 3D feature points. The camera transformation is appended to the scene transformation when rendering the virtual geometry allowing the tree to appear as if it was part of the real world.

For each frame, a 2D Delaunay triangulation of the feature points is computed to construct a screen space mesh representing scene geometry. An incremental algorithm [Lischinski, 1994] is used to create the triangulation, and a quad-edge data structure [Guibas and Stolfi, 1985] is used to store and traverse the mesh. The mesh is constructed in 2D screen space but depth information from the 3D feature points is used when rendering. This mesh is rendered with the current video frame as a texture providing a background to overlay the virtual tree rendering. This allows for (approximate) occlusion handling between the real scene and digital objects.

The tracking system is first initialized by pointing the camera at a roughly-planar textured surface, then moving the tablet forward and backward and rotating it around the

surface. This allows the system to identify an initial set of feature points, which determines the virtual ground plane. I use a RANSAC method [Fischler and Bolles, 1981] to fit a plane to these initial feature points. The centroid of the feature points, projected onto the plane becomes the origin of the virtual world. This entire process takes a few seconds, after which a seedling is planted at the origin of the world and the modeller can start creating a tree.

5.4.2 Interface

In TreeSketch, the main method of interacting with the tree is through brushing to induce growth. Brush strokes are projected onto a camera facing plane with the scene depth controlled by the starting point of the stroke. This interface allows the camera to be freely panned and zoomed by moving the tablet. While painting, a constant scene depth is maintained between the camera and the plane of the brush. This allows the brush depth to be conveniently modified throughout the stroke by moving the tablet forward and backward. Larger strokes are made possible by panning the tablet while painting.

5.4.3 Interaction with the Environment

The environment in which the tree is grown plays an active roll in its development. While modelling, the tree can be pruned to the scene geometry resulting in a topiary effect where all branches intersecting real world objects are cut. Alternatively, the environment can play a constant roll in the tree development by restricting attraction points to exist strictly in front of all scene geometry. The latter method results in more natural'. looking tree forms which appear to grow into the environment instead of being cut to fit but does not strictly guarantee that the tree will not grow through sections of the environment.

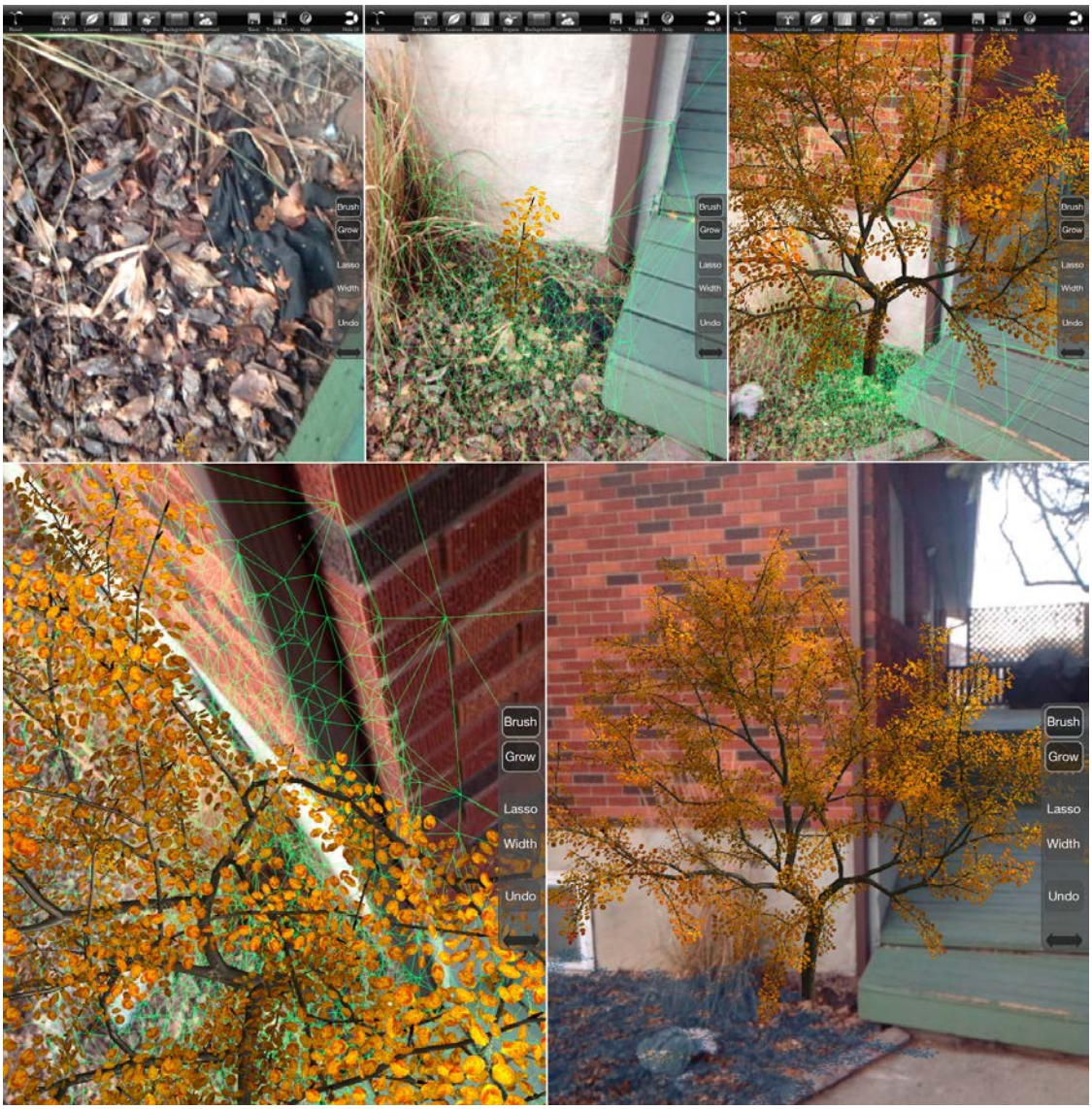


Figure 5.21: From left to right, top to bottom. The scene is initialized on a highly textured planar section of ground. A seedling is sketched using the brush, feature points (vertices) and the corresponding depth map are visualized as a green mesh. The seedling is grown for a few stages then pruned to the depth of the scene resulting in geometry that grows around the wall instead of intersecting it. A view from above the tree, notice how the tree is still growing around the edge of the wall but no branches are growing through it. Final shot of the tree from a further out view.

5.4.4 Discussion and Limitations

Modelling a tree at a correct scale and fitting within the constraints of an environment is difficult when the tree is not presented in context. The augmented reality interface alleviates this issue for trees that are being modelled for incorporation in a real environment (i.e. for architectural visualization). This interface allows for existing trees to be recreated digitally on site while allowing the modeller to compare the two structures.

A suitable environment is required for initializing and maintaining accurate tracking. Surfaces that are highly reflective, transparent or lack texture do not allow for robust tracking. Fortunately, outdoor environments are highly textured and often do not have large reflective or transparent surfaces making tracking outdoors remarkably robust.

5.5 Model Visualization

Trees modelled with TreeSketch are intended to be assembled into scenes and rendered using external programs. However, a relatively realistic rendering during the modelling process helps reduce the number of iterations, providing the modeller with a better impression of the final tree appearance. To this end, the following rendering techniques are combined:

- 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 modeller 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 with the shadow propagation algorithm as an inherent component of the growth model. This information is used 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 (Figure 5.22 left).

- Shadows. Shadow-mapping is used to cast Gaussian-filtered blurred shadows on the ground and hard shadows within the tree [Eisemann et al., 2011].

Figure 5.22: Examples of complex tree structures modeled and rendered with TreeSketch.

Figure 5.22 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. In addition, differences in leaf color are accounted for by shifting the hue of leaves on less vigorous branches towards yellows and reds (leaf placement and orientation are discussed in Appendix C). The magnitude of this shift is a parameter controlled by the modeller. 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 modeller is not interacting with the system.

5.6 Results

Diverse trees modeled with TreeSketch are shown in Figure 5.23. 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 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 5.23: Diverse trees modeled using TreeSketch. Figure from Longay et al. [2012].

Figure 5.24: A scene with trees generated using TreeSketch. The scene was assembled and rendered in Maya. Figure from Longay et al. [2012].

Figures 5.24 and 5.25 provide examples of generating realistic tree models that address the artistic requirements of target scenes. Figure 5.24 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 5.25.

Figure 5.25: A scene with trees generated using TreeSketch. The scene was assembled and rendered in Maya. Figure from Longay et al. [2012].

5.7 Discussion

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, the statements made about TreeSketch are supported by positive feedback from the users of three public releases and over 50 000 downloads. Users stated 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.

Crucial model parameters are manipulated through a set of interactive widgets which have been specially designed for the purpose of tree modelling. These widgets greatly help in navigating the space of architectural characteristics of the models, especially when manipulating correlated variables such as direction and magnitude of gravimorphism, or the different factors affecting directions of growth. Visual feedback of changing parameters is provided by a two-dimensional schematic tree model which changes form instantaneously as the modeller manipulates the widgets.

Modeless design is essential to the TreeSketch interface where main operations are instead distinguished by the number of touch points and context within the scene. When necessary, secondary editing operations such as editing of branch widths are performed by enabling a spring-loaded mode (having the modeller press and hold a button while performing the gesture). By utilizing multi-touch to enable these spring-loaded modes, the act of pressing and holding the button simply feels like a component of the gesture, maintaining the seamless impression of the interface. Multi-touch is particularly useful when bending branches and offers a method for controlling brush size or rotating the tree while painting. Opinions of users strongly indicate that the touch and gesture-based interface significantly enhances the interactive modelling process.

The current design of TreeSketch is the result of iterative refinement and incorporation of user feedback. The initial version utilized the two different growth models that were presented by Pałubicki et al. [2009]. The user was able to freely intermix interactive painting with a brush or grow the tree autonomously, however, each growth mode produced

branches with a different character. Moreover, if the modeller grew the tree autonomously after brushing many branches would be shed, possibly even those explicitly specified. This motivated the development of the hybrid growth model presented in Chapter 4 which results in a consistent branch character and significantly reduces branch shedding. The need for an interactive widget to control the direction of branch growth was also motivated through user feedback. The initial version used two sliders to control the relative strength of branch straightness and tropism, however, it was not clear that these parameters were interdependent. The triangle widget makes this coupling clear as the user can see visually how increasing the strength of one parameter decreases the other two.

Chapter 6

Trees To Forests

Large landscapes of trees can be generated using a small number of tree models that are *instanced* and independently transformed to provide the illusion of diversity. The resulting landscapes are visually acceptable when viewed from a distance, but fail when the viewer is close. A key artifact with this technique is the lack of crown plasticity. Since the same tree geometry is used, it does not fit with its neighboring trees. This either results in unrealistic gaps between trees or intersecting geometry.

Instead of generating plant models independently and later combining them into a scene, growth of a population of plants can be simulated simultaneously. Algorithms that generate the form of trees by simulating competition are particularity well suited for this as plants can compete for a shared resource. Simulation from this perspective dates back to the seminal work of Greene [1989] and Takenaka [1994] and has also been utilized in the more recent work of Měch and Prusinkiewicz [1996], Rodkaew et al. [2003] and Pałubicki et al. [2009]. In these models, neighboring plants compete with each other for space or light, which results in non-uniform, adaptive tree shapes.

However, such simulations do not scale to the level of entire landscapes. Since the growth of one tree has the potential to impact the form of all other trees, the entire ecosystem must be simulated together. Moreover, tuning parameters to obtain realistic results from a simulation of multiple species with varying growth rates is challenging. Faster growing trees easily overtake slower growing models. Trees with a drooping character will take significantly more simulations steps to reach a desired height than trees with branches

tending upwards. This must be accounted for if two trees are grown together in the same environment and changing the parameters of one species can require recalibrating all species. Pałubicki et al. [2009] showed a scene with two trees of different species growing in the same simulation but their interaction was limited by the placement of a house between them.

Multi-level techniques have proven useful in generating large complex landscapes [Deussen et al., 1998]. Lane and Prusinkiewicz [2002] propose a system in which a distribution of tree bases is computed by simulating interactions between tree species such as clustering or inhibition. This distribution defines the placement of predefined crown shapes which are substituted for tree geometry when the tree is rendered. I extend their model by introducing a brush-based interface for distributing trees and simulating competition for space between trees at the level of tree crowns. The resulting shapes act as constraints for self-organizing trees that grow into the space resulting in a natural form. Since inter-tree competition is simulated at a higher level than the trees themselves, the detailed form of each specific tree can be generated independently, given its constraining shape. Therefore, there is no need to tweak parameters to account for growth speed of a particular species. This allows parallel computing techniques to be utilized, significantly increasing the scalability and performance of the algorithm. Growth parameters of individual trees can be tweaked without the need to regenerate the entire landscape. This eliminates the concern of large scale changes when editing a single tree model.

6.1 Algorithm Overview

I propose to generate landscapes of trees using a multi-stage process.

- The terrain is defined using a brush-based interface to specify its elevation and areas where trees are excluded from being placed.
- On this terrain, the distribution of tree bases is defined interactivity with a brush. The algorithm proposed by Lane and Prusinkiewicz [2002] is used to maintain inter-species interactions, such as clustering and inhibition.
- From this distribution, trees compete for crown space. This segments the entire space into volumetric-regions associated with each tree.
- Detailed models of individual trees are generated using the growth model defined in Chapter 4. Attraction points are restricted to be within the volumetric-region associated with each tree. Trees can be simulated in parallel as they no longer depend on each other.
- Growth parameters for each tree can be adjusted by the modeller. Only the adjusted tree needs to be updated and the rest of the landscape remains unchanged.

6.2 Species Definition

Each tree species is defined by a set of growth parameters (Chapter 4) and a *representative tree shape*. The tree shape is made up of two components, the trunk and crown. The trunk is a cylindrical component defined by its width and height (T_w, T_h) and the crown is an ellipse vertically offset by the trunk height and defined by the length of its major and minor axes

(C_w, C_h) (Figure 6.1). The shape of the tree crown is created such that it resembles the size of a free-standing tree model grown with the corresponding parameters. The dimensions of this shape, modified by a per-instance scale factor, define the maximum height of the tree and its projected crown area $(\pi \frac{C_w}{2}^2)$.

Figure 6.1: Example representative crown shapes

6.3 Terrain Specification

The initial size of the plot of terrain is specified as a parameter to the system. The terrain, represented internally as a height-field, is initialized to a planar surface offset slightly by Perlin noise. The modeller can change the elevation of the terrain by brushing which moves the region within the brush up or down. If the height at any point is below a predefined water level, an animated plane of water is exposed (Figure 6.2). The modeller can also brush regions of the terrain which they wish to exclude from tree placement. These regions are rendered with a rock texture to provide visual feedback.

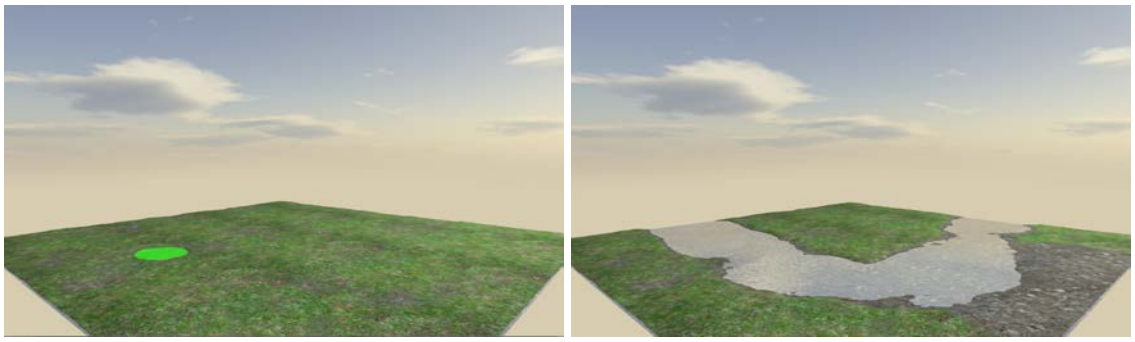


Figure 6.2: An example of brushing the terrain. Left) The initial state, modeller brush shown in green. Right) A region of terrain has been brushed below the water level creating a river. The region on the right bank (shown by a rock texture) has been excluded from tree placement.

6.4 Tree Distribution

The ecosystem modeling method presented by Deussen et al. [1998] uses (among others) the process of self-thinning to generate the locations of plants of various species. The self-thinning process initially generates a relatively dense distribution of small plant seedlings. Then, through an iterative process, the simulation grows plants radially at species-specific rates, and some plants are removed when they are *dominated* by a larger plant (Figure 6.3).

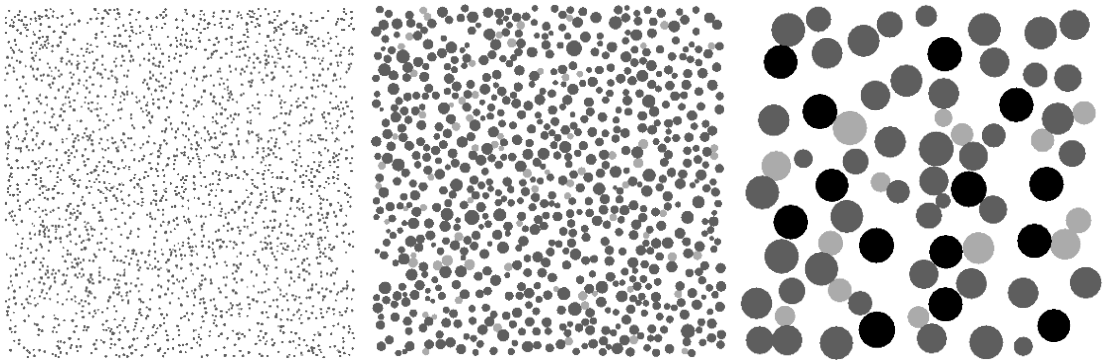


Figure 6.3: Distribution of plants created using a self-thinning model. Plant locations represented by light grey circles are being *dominated* by a larger plant. Figure from Lane and Prusinkiewicz [2002]

While this method can produce realistic distributions of plants, it lacks control over the clustering of the distribution. A model proposed by Lane and Prusinkiewicz [2002] creates clustered distributions by generating new plant *seeds* in the vicinity of large plants of the same species. This seeding is intertwined with the process of removing dominated plants. Lane and Prusinkiewicz [2002] noted that in the resulting distributions, it is probable to find a plant of a particular species in the vicinity of another plant of the same species. This observation motivated the development of a simple and more controllable statistical method of creating plant distributions, the deformation-kernel method [Lane and Prusinkiewicz, 2002]. In contrast to the models based on self-thinning, the deformation-kernel method iteratively adds trees to the distribution instead of removing them from an over-dense set. This is highly desirable for interactive applications as adding more plants will never result in the removal of existing ones. The following section presents this method in detail.

6.4.1 The Deformation-Kernel Method

The deformation-kernel method maintains a probability field for each species. This field defines the probability of placing a new plant of a particular species at each location in the field. Adding a plant to the distribution at location $P(x,y)$ modifies the probability fields of all species by multiplication of the corresponding kernel function at location P in the probability field.

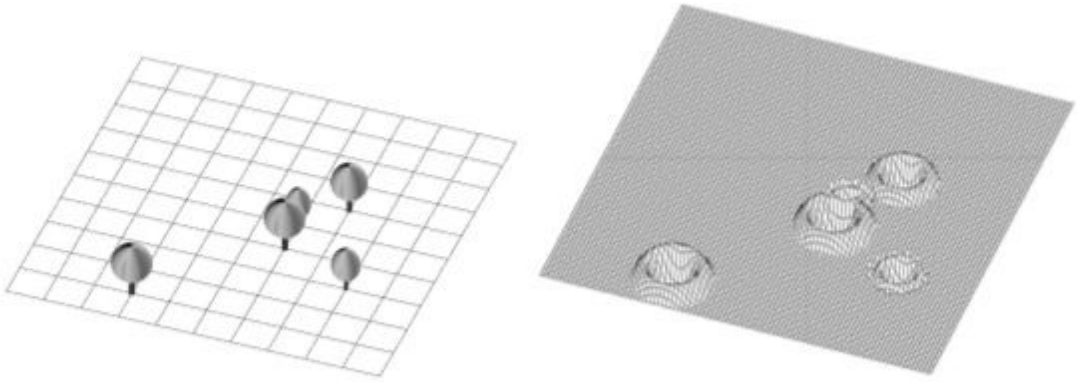
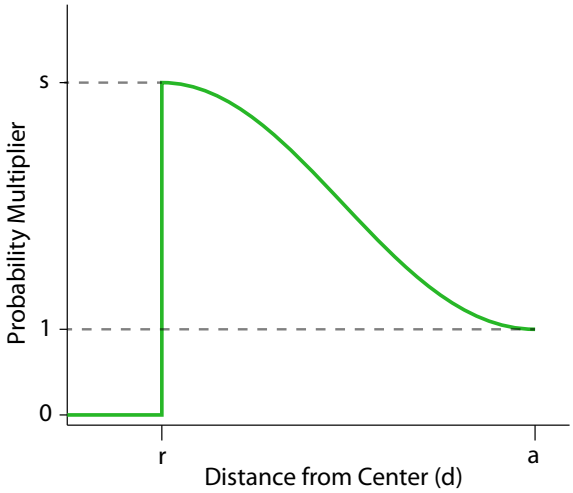


Figure 6.4: Visualization of the probability field as a height map after the placement of a few trees. Figure from Lane and Prusinkiewicz [2002].

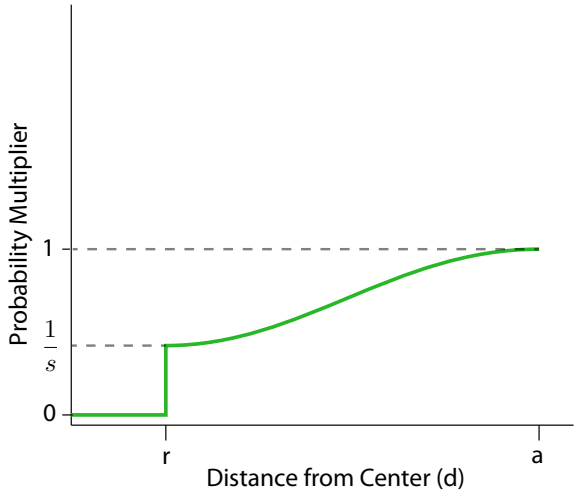
Kernel Functions

The pairwise interactions between species are defined using kernel functions that represent the probability of a new tree being placed at a distance d from the location of an existing tree. All trees occupy some radius r from their central position where they obstruct placement new trees ($\frac{T_w}{2}$, Section 6.2). Within an active radius, a, trees can effect the probability of new tree placement. Clustering of tree locations can be simulated by increasing the probability of new trees being placed in the vicinity of existing trees, using the following kernel function:



$$K(d) = \begin{cases} 0 & : d \leq r \\ (s-1)(6x^5 - 15x^4 + 10x^3) + 1, \quad x = 1 - \frac{d-r}{a-r} & : r < d \leq a \\ 1 & : d > a \end{cases} \tag{6.1}$$

The smooth-step function of Ebert et al. [2003] is used to blend from s to 1. The exact form of this function is not important, any ease in-out function such as Hermite interpolation could be used. The inverse effect is obtained using an inhibitory kernel which decreases the surrounding probability and is defined as follows:



$$K(d) = \begin{cases} 0 & : d \leq r \\ 1 - \frac{1}{s} + \frac{1}{s}(6x^5 - 15x^4 + 10x^3), \quad x = \frac{d-r}{a-r} & : r < d \leq a \\ 1 & : d > a \end{cases} \tag{6.2}$$

The form of these equations allows parameter s to scale the *effect* of clustering or inhibition, even though it is inversed in their functions. For each pairwise species interaction, the modeller selects which kernel is to be used and defines its scale and active range. Lane and Prusinkiewicz [2002] provided support for a wider range of kernels, specifically those which did not include the occupation radius r. This was necessary in their system because they used static crown shapes to constrain the growth of the trees and neighboring plants did not compete for crown space. To obtain a natural looking distribution without gaps between neighboring trees, they allowed for intersecting tree bases. Since I simulate trees competing for crown space, there is no need to have intersections.

Algorithm

Each probability field is implemented as a two dimensional array of floating point numbers, with all values initially set to 1. The corresponding kernel function is multiplied into each field each time a new tree is placed in the distribution.

The position of each tree, $P(x,y)$, is computed as follows: The sum of each row in the species probability field is calculated and stored in an array R. The values of R are adjusted sequentially, $R[i] = R[i] + R[i-1]$ to create a monotonically increasing sequence of values. A random number, k, is then generated between 0 and the value stored in $R[Max]$. The y component for the plant can then be computed by first finding the row, r, such that $R[r] \leq k < R[r+1]$ and then linearly interpolating between $R[r]$ and $R[r+1]$. Figure 6.5 illustrates this process. Since rows containing higher probabilities map to a larger range, a bias is created.

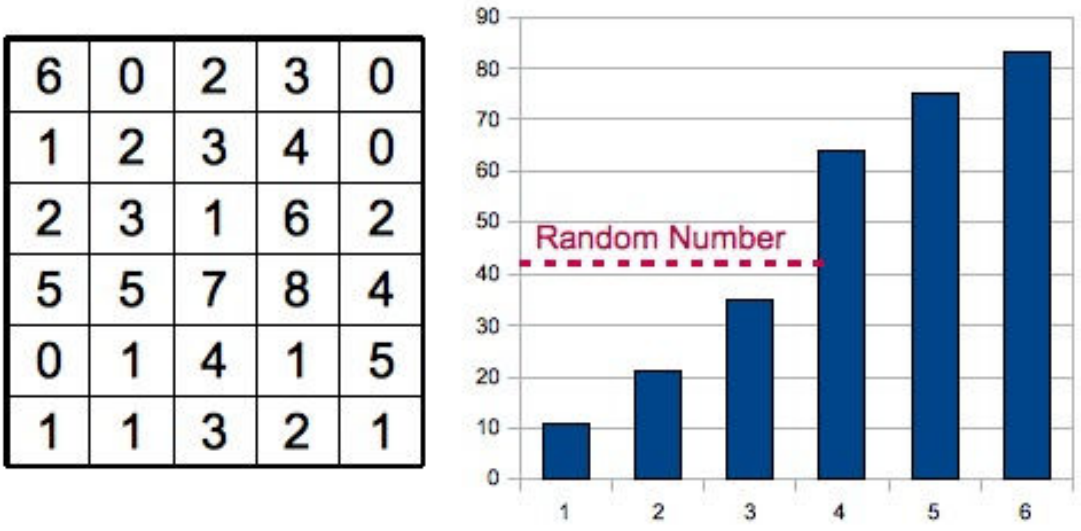


Figure 6.5: A hypothetical probability field, left, and resulting monotonically increasing function of the sums of rows, right.

The x component of the tree position is computed in a similar manner. The sum of all columns in the row, r, is stored in an array C. The values of C are adjusted as above to create a monotonically increasing sequence. A random number, n, is then generated between 0 and the value stored in $C[Max]$. The x position for the plant can be computed by first finding the column, c, such that $C[c] \leq n < C[c+1]$ and linearly interpolating their positions. With the position of the new plant calculated, corresponding kernel function is multiplied into the probability field of each species at the location of the tree. Figure 6.4 shows these effects.

The following figures show sample distributions created with the algorithm:

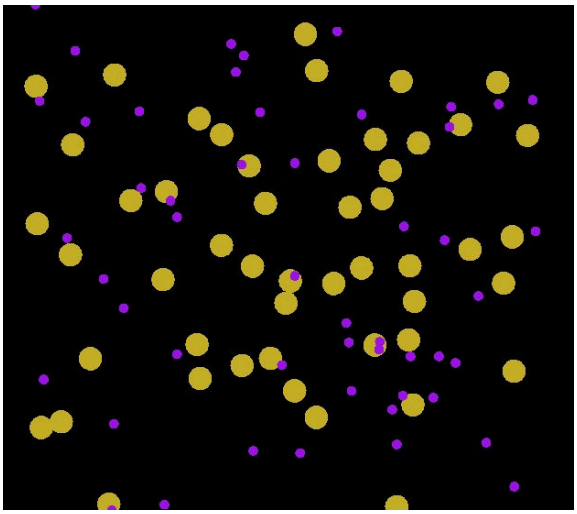


Figure 6.6: An example of a random distribution.

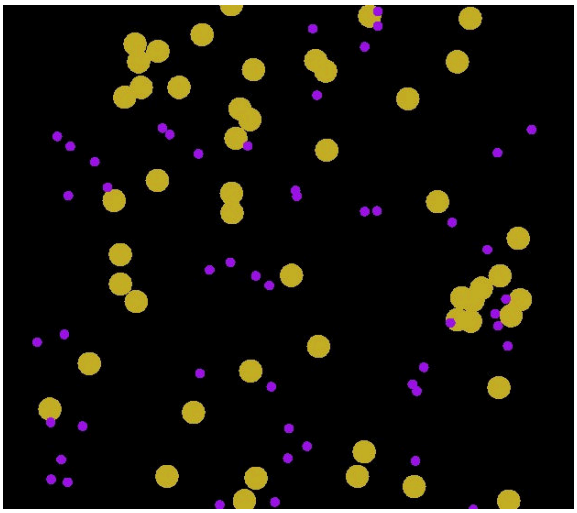


Figure 6.7: An example distribution showing clustering between plants of the same species. No interactions between plants of different species.

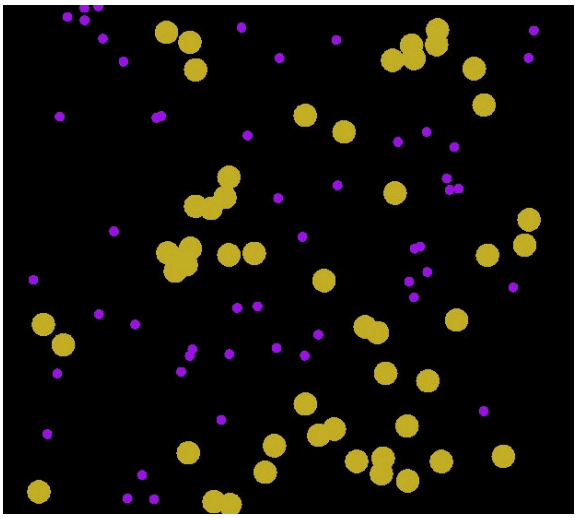


Figure 6.8: An example distribution showing clustering between plants of the same species. An inhibitory interaction between plants of different species.

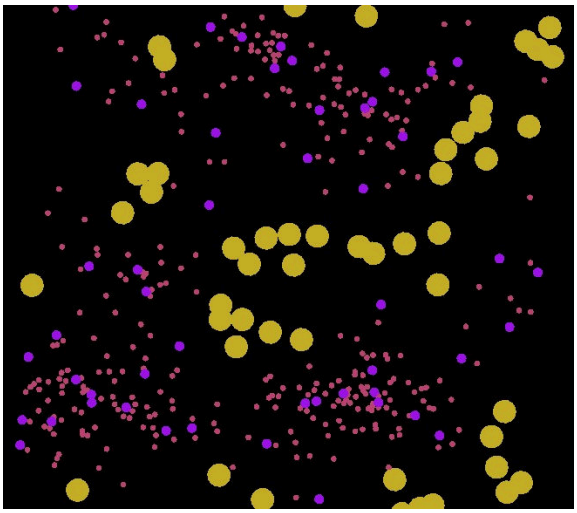


Figure 6.9: A distribution with three species. Yellow trees cluster around their own species but inhibit all other species. Purple trees inhibit yellow and other purple trees but have a clustering interaction with the pink species. The pink species does not effect the others.

6.4.2 Interface

The algorithm presented in the previous section is used to iteratively add trees to the landscape. However, it is not sufficient to have the trees simply scattered across the terrain. Therefore, I propose a brush to define the placement of trees.

A tree can either be an instance of a predefined species (Section 6.2), or a *hero* tree. A hero tree is one that has been pre-designed by the modeller, specifically for the scene, using a tool like TreeSketch (Chapter 5). Available tree types are presented in a bar on the left side of the screen (Figure 6.10). Each tree type T_i is associated with a relative probability $P_i, 0 \leq P_i \leq 1$ set by the modeller using the side bar (Figure 6.10).

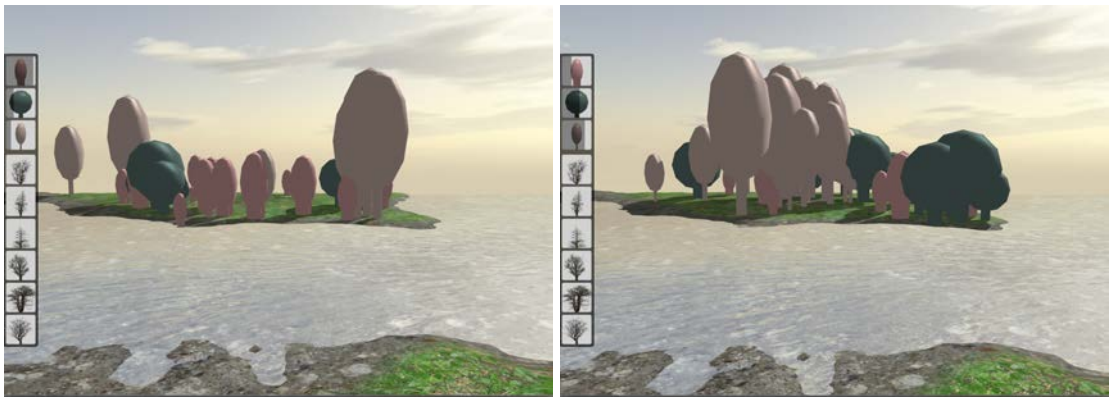


Figure 6.10: Available tree types and their relative probabilities are defined in the species bar on the left side of the screen. The top three thumbnails represent predefined species while the bottom six are hero trees. The magnitude of probability is represented by a transparent grey overlay on the species thumbnail. The modeller can adjust the probability of a species by dragging left or right on its thumbnail. Left) Trees are brushed on the island favouring the smaller pink species. Right) Species probabilities are adjusted, clustering of species is evident.

The radius and magnitude of the brush can be adjusted by the modeller. When the modeller is brushing on the terrain, trees are added iteratively at a rate proportional to the brush magnitude. Each iteration, a tree type is chosen using the following method (similar as above): The probabilities of each tree type are stored in an array P such that $P[i] = P_i$. The values of P are adjusted sequentially, $P[i] = P[i] + P[i-1]$, to create a monotonically increasing sequence. A random number, n, is then generated between 0 and the value stored in $P[Max]$. The tree type corresponding to the probability $P[t]$ is found such that $P[t] \leq n < P[t+1]$. An instance of this tree type is generated and a scale factor is randomly computed between a modeller defined min and max scale.

The method discussed in the previous chapter is used to compute the position of the tree instance with the following modifications. The circular area of the brush is used as a

mask such that only points of the probability field within the brush are considered for tree placement. Similarly, tree distribution is also masked by the terrain where areas specifically excluded by the modeller (Section 6.3) and areas below the water level are not considered.

This algorithm is fast enough to provide the feeling of scattering trees in real time. With a small brush radius the modeller can explicitly place a tree. Using a larger brush radius, the effects of clustering and inhibition play an important role (Figure 6.10). The modeller can tweak the resulting distribution by explicitly moving, or scaling any tree interactively (Figure 6.11).

Figure 6.11: Instances of tree species are visualized using their representative tree shapes. Hero trees (here imported from TreeSketch) are shown in full geometry. Hero trees are explicitly placed and scaled. Smaller pink trees are scattered with a brush along the river bank. Larger species are brushed in the background.

6.5 The Neighborhood Effect

One of the fundamental characteristics controlling tree form, as identified by Harper [1977], is the neighborhood effect. This refers to the adaptive shape of tree crowns in the presence of neighbors, also known as crown plasticity (this effect can be seen in Figures 6.12, 6.26 (bottom), 6.27 (bottom) and 2.4). In order to grow trees independently and still obtain realistic results, it is important to capture this effect.

Since the seminal work of Mitchell [1975], much forestry research has focused on simulating the shape of tree crowns. The model of Sorrensen-Cothern et al. [1993] expanded on earlier models by introducing interaction between trees in the vertical space using stacks of discs to represent the general shape of conifer tree crowns, and adjusting for intersections with neighboring trees to determine more realistic shapes. Building on this work, the model of Grote and Pretzsch [2002] simulates stacks of *growing* discs where the growth rate of each layer is specifically tuned and calibrated to match a particular species. The goal of these simulators is to obtain a crown shape which can be calibrated by experimental data for a particular species and used to study other aspects of forest dynamics.

At the edge of the forest or on the bank of a river, trees can be found leaning into open space and away from their neighbors. Hallé et al. [1978] termed this behavior the *river-bank effect*, and it can be viewed as an extreme case of crown displacement. While free-standing trees may have their crowns directly centered over their trunk bases, in the presence of neighbors, tree crowns are often displaced in a particular direction [Brisson, 2001, Muth and Bazzaz, 2002, 2003, Schröter et al., 2012]. The size and proximity of a neighboring tree provide the best indication in determining the strength of crown displacement away from the particular neighbor [Muth and Bazzaz, 2003]. For trees with a relatively symmetric

neighborhood, the effect is less noticeable as these *neighbor pressures* cancel each other out [Brisson, 2001]. However, as neighborhood asymmetry increases, so does the magnitude of crown displacement [Muth and Bazzaz, 2003].

Figure 6.12: A stand of trees tilting away from their neighbors at the edge of the forest. Image from https://flic.kr/p/e3ySGX.

In this section I present an algorithm which simulates competition between trees for crown space. This process segments space into volumetric regions associated with individual trees. As mentioned in Section 6.1, these volumes constrain the growth of the tree. It is not necessary for these volumetric regions to look like realistic crown shapes, only to capture the higher level interactions between species which result in crown plasticity.

6.5.1 Space Constraints

The main cause of crown plasticity is competition between neighboring trees. I present an algorithm to segment the space such that the resulting volumetric regions exhibit the effects of this competition. A Voronoi diagram is commonly used to divide space into regions, where all points in a given region are closer to the *seed* point for the region than to any other seed. Figure 6.13 shows an example of a Voronoi diagram constructed from a tree distribution. As seen in this figure, the size of the tree has no effect on the size of its corresponding region. Some small trees are allocated too much space, while larger trees are too constrained. Moreover, all dividing lines between trees are straight which results in visible artifacts if these shapes are used to constraint tree growth.

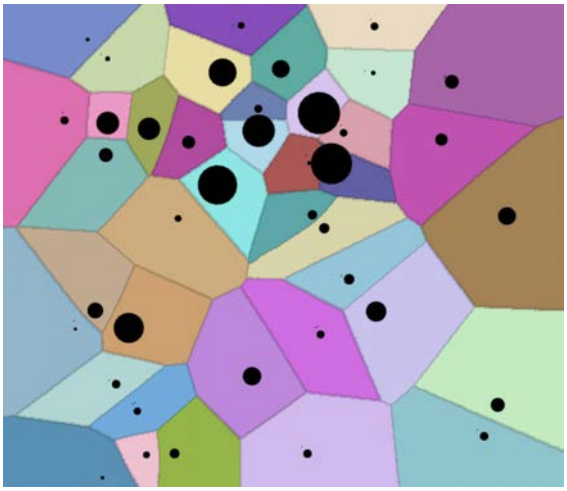


Figure 6.13: A Voronoi diagram constructed from positions of trees. Tree positions are shown by black circles with radius proportional to the projected area of their representative tree shape (Section 6.2).

Inspired by the idea of generating crown shapes from independently growing discs [Grote and Pretzsch, 2002], I simulate growing and colliding regions centered at the base

of each tree. The space of tree growth is divided into a two dimensional grid, each cell containing the ID of its *claiming* tree. Each tree initializes a discretized circle under its position, with a radius proportional to its trunk radius ($\frac{T_w}{2}$, Section 6.2), and claims all grid cells within. The rate of growth for each circle is proportional to the projected area of the representative tree shape (Section 6.2).

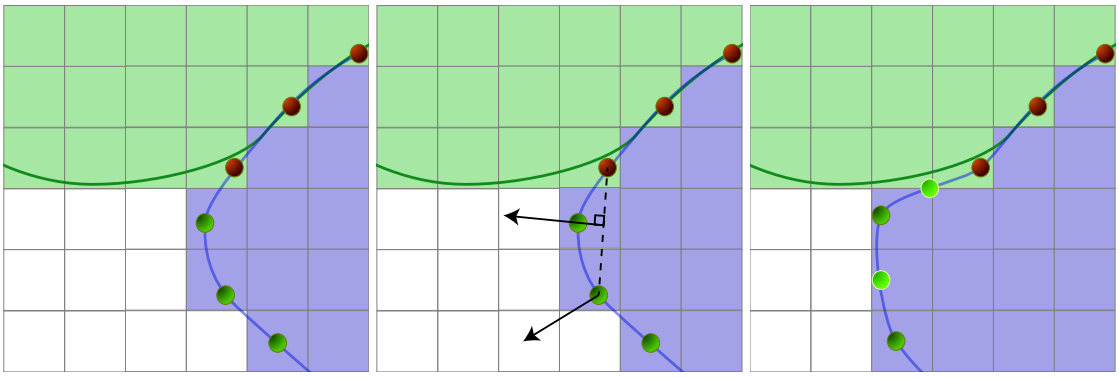


Figure 6.14: Stages of the boundary propagation algorithm. Left) Growth of the blue area halted for the points which reside in grid cells claimed by a different tree (red). Center) The direction of growth for each point is defined as the direction perpendicular to the vector between its two neighbors. Right) Points are moved to their new location and new points (bright green) are added on line segments longer then a threshold length.

At each step of the simulation, each point grows in the direction perpendicular to the line between its two neighbors (Figure 6.14). After growth, if a point is located in an unclaimed grid cell, it claims it. Points that reside in a grid cell claimed by another tree halt growth. Handling collisions on this coarse grid structure results in a fast and robust implementation. As regions grow, new points are added between any two points which are further apart than a threshold distance. The simulation stops when all grid cells have been claimed. Figure 6.15 shows key frames of this simulation.

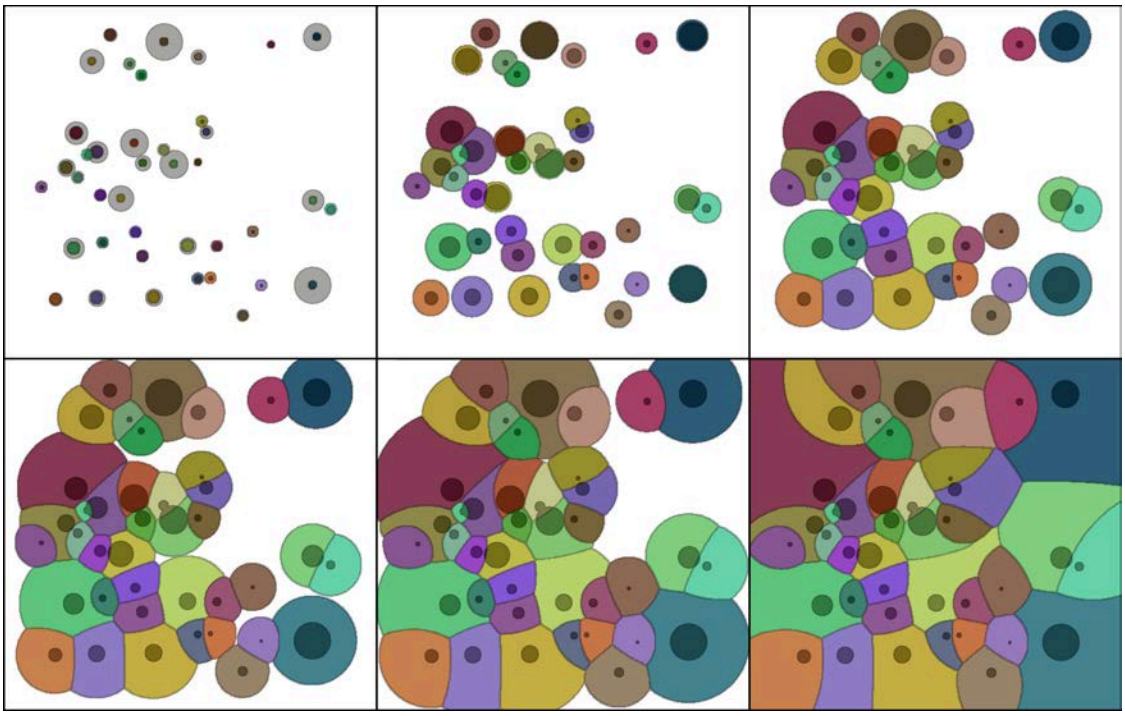


Figure 6.15: Frames of the boundary propagation algorithm. Simulation time increasing from left to right, top to bottom. Semi-transparent grey circles represent tree locations with a radius proportional to the projected crown area (Section 6.2). Larger trees have a faster growth rate and are able to claim more space.

The resulting structure is similar to a multiplicatively weighted Voronoi diagram, except that by simulating collisions between regions, I ensure their connectedness. This adaption has been used as a model of growing crystal formations [Schaudt and Drysdale, 1991]. As compared to a standard Voronoi diagram (Figure 6.16), boundaries between regions are curved and larger trees able to claim significantly more space.

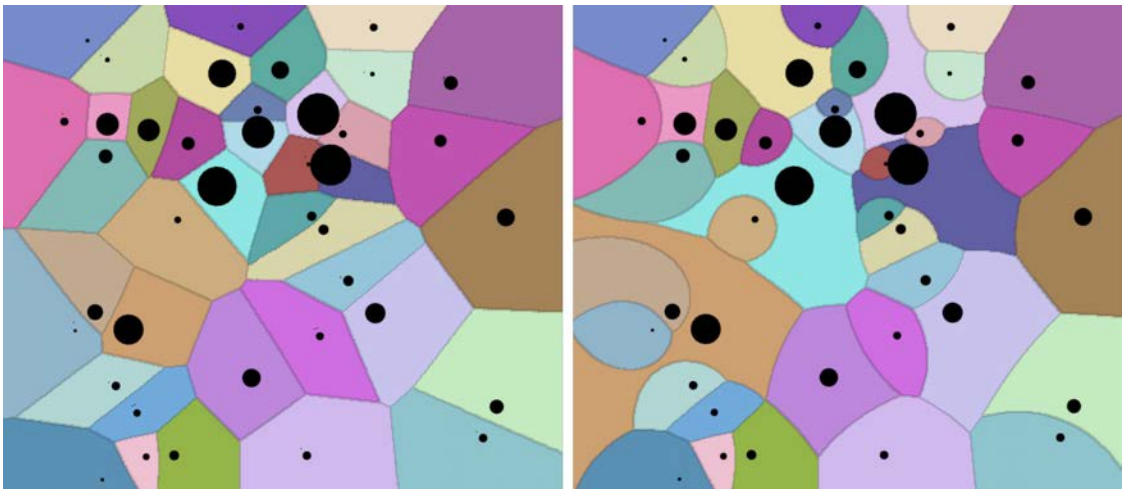


Figure 6.16: Comparison between a standard Voronoi diagram (left) and a diagram generated from boundary propagation (right). Tree positions are shown by black circles with radius proportional to the projected area of their representative crown shape.

Yet, this diagram only represents the situation at the base of trees. Extruding this diagram upwards would not take into account the height of trees and not allow for branches of a taller tree to overhang a shorter neighboring tree. I extend the simulation into 3D by simulating this process in multiple layers. Instead of a grid structure dividing the two dimensional space, I employ a voxel space dividing three dimensional space. Each layer of the voxel space considers only those trees whose representative tree shape extends beyond the height of the layer (Section 6.2). Since the regions only grow in two dimensions, each layer can be simulated independently using parallel computing techniques. Figure 6.17 shows slices of the resulting voxel space.

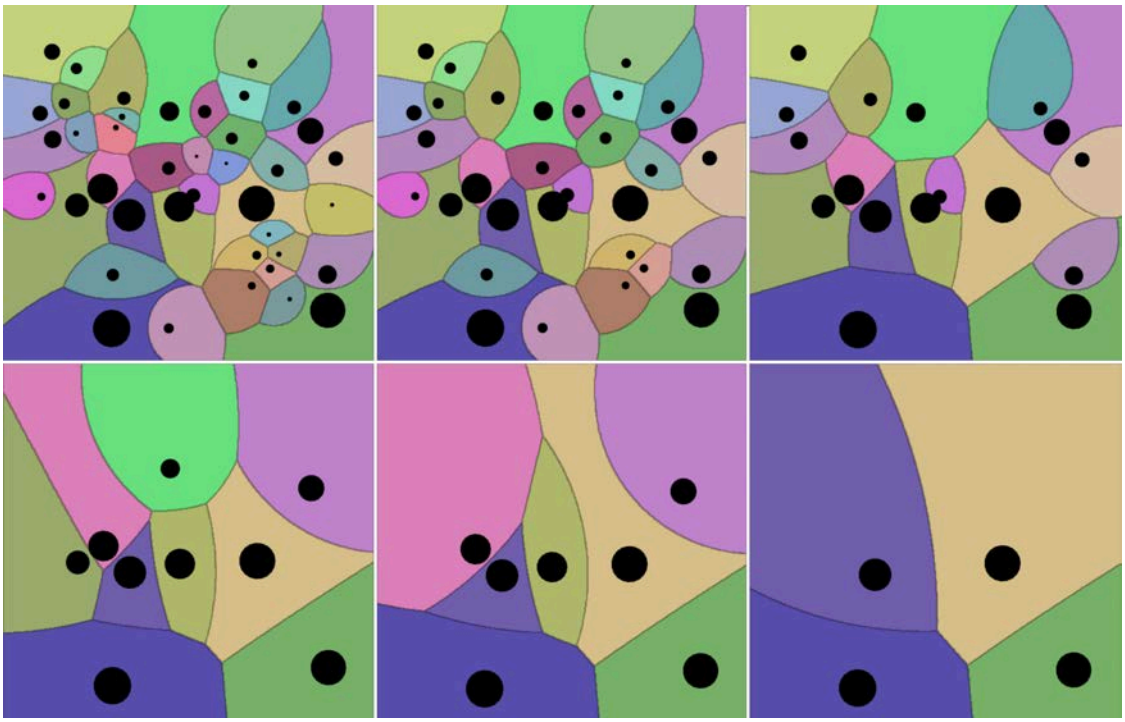


Figure 6.17: Slices of the constraint voxel space increasing in height, left to right, top to bottom. Radius of circles is proportional to the height of their corresponding tree shape (Section 6.2).

Since the final crown shape of hero trees is already known, they do not take part in this process. Hero trees are instead incorporated afterwards by traversing their tree structure and claiming all voxels which contain a hero tree segment. The final regions associated with each tree are used to constrain tree growth as shown in Section 6.6.

6.5.2 Neighbourhood Asymmmetry

In the case where the tree exists in an asymmetric neighborhood (e.g. on the bank of a river), I quantify the direction and magnitude of asymmetry in a *Neighbourhood Asymmetry Vector*.

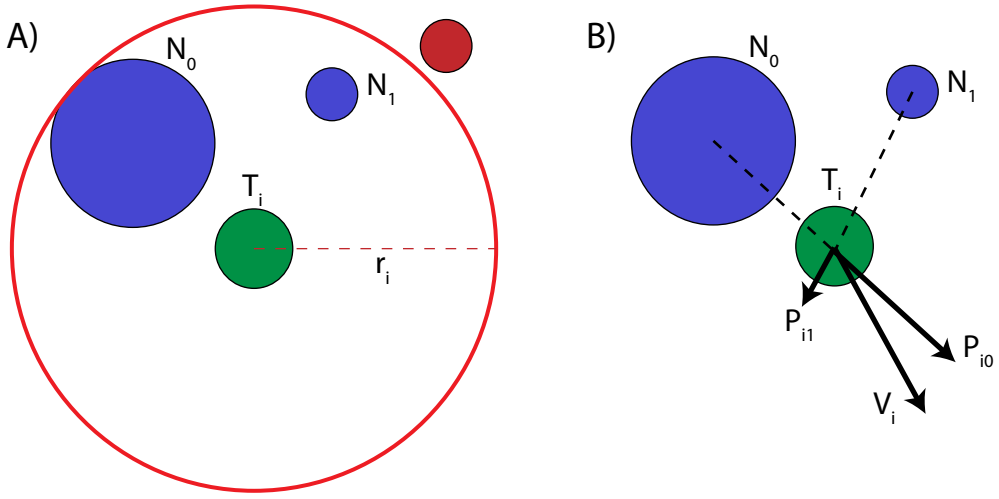


Figure 6.18: Computing the neighborhood asymmetry vector. A) Trees within the radius r_i (blue) are considered neighbors and influence the tree. B) Neighbor pressure P_{ij} is weighted by the relative area and distance of the neighbor. The sum of all neighbor pressures defines the neighborhood asymmetry vector V_i.

As shown in Figure 6.18, the neighborhood asymmetry of a tree can be modeled as a weighted sum of vectors from neighboring trees [Brisson, 2001, Muth and Bazzaz, 2003]. For each tree, T_i, I define its neighborhood, $\mathcal{N}_i$, as all trees within the radius r_i from the tree base location. The vector P_{ij} represents the neighbor pressure from tree $T_j \in \mathcal{N}_i$ on T_i and is defined as:

$$P_{ij} = \omega_{ij} * \frac{T_i - T_j}{||T_i - T_j||}, \qquad \omega_{ij} = \alpha_a \frac{A_j}{A_i} + \alpha_d \left(1 - \frac{||T_i - T_j||}{r_i}\right) \tag{6.3}$$

where the weight ω_{ij} is proportional to the relative size of the trees, measured in projected crown area $A_{i,j}$ [Muth and Bazzaz, 2003], and inversely proportional to the distance between them. The parameters $\alpha_{a,d}$ allow for tuning the relative importance of each ($\alpha_a = 1.0, \alpha_d = 2.0$ for results shown in this thesis).

An important factor in determining a tree's ability to displace its crown is whether or not it has sufficient space. The maximum extent of tree growth is determined by the volumetric region computed in the previous section. I compute the centroid C_i, and bounding-circle radius B_i, for each layer of the region. If a tree base is near the edge of its volume then it will have sufficient space to displace its crown, and its distance to the centroid relative to the radius of the region will be 1. If the tree is positioned directly over the centroid, it does not have enough space to displace its crown and the distance to the centroid relative to the radius will be 0. The sum of all neighbor pressures scaled by this ratio defines the neighborhood asymmetry vector:

$$V_i = \frac{||T_i - C_i||}{B_i} \sum_{j \in \mathcal{N}_i} P_{ij} \tag{6.4}$$

Figure 6.19 shows the resulting neighborhood asymmetry vectors for two sample distributions. This vector is used to bias the growth of the tree as shown in the following section.

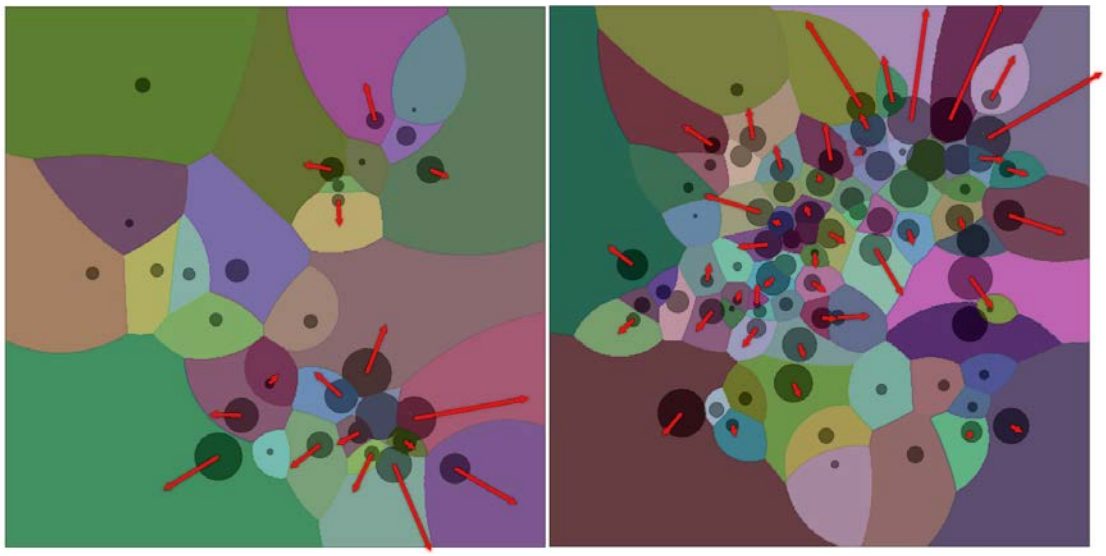


Figure 6.19: Neighborhood asymmetry vectors (red arrows) overlaid on the constraint graph. Positions of trees are shown as semi-transparent black circles with a radius proportional to the projected crown area of their representative tree shape. Trees at the center of clusters are not affected while trees on the edges of clusters have strong neighborhood asymmetry, especially when neighboring large trees.

6.6 Tree Growth

As stated in Section 6.1, this algorithm for generating landscapes is a multi-stage process. The position and species of each tree instance is defined by the distribution algorithm discussed in Section 6.4, while spatial constraints and vectors of neighborhood asymmetry are computed as in Section 6.5. At this stage, growth of each tree can be simulated independently using the autonomous growth algorithm discussed in Chapter 4. Attraction points are scattered around bud locations, but removed if they land outside of the voxel constraints. The maximum height of a tree h_{max}, is defined by the instance scale and the

height of its representative tree shape ($S_i * (T_h + C_h)$). Growth of the tree is halted when a branch reaches the maximum height. Further growth would allow the tree to fill its crown constraint shape but would result in an unnatural appearance near the boundary (i.e. a flat top).

The neighborhood asymmetry vector V_i, is incorporated as an additional tropism. This helps to push the growth of the tree away from its dominant neighbors and provides the characteristic tilting of the main axes as seen in trees at the edge of a forest. In order to effect only the larger branches of the tree, the effect of this tropism fades out as the tree grows. The final tropism direction F_i is defined as:

$$F_i = (1 - \alpha) * (V_i + S_t) + \alpha * S_t, \qquad \alpha = \left(\frac{h}{h_{max}} \right)^{\frac{1}{2}} \tag{6.5}$$

where S_t is the direction of tropism defined by the species parameters. The exponent in α concentrates this effect in the early stages of development. Figure 6.20 shows a comparison of trees grown with and without this additional tropism. Figure 6.21 shows how larger trees remain unaffected while smaller trees tilt away from them.

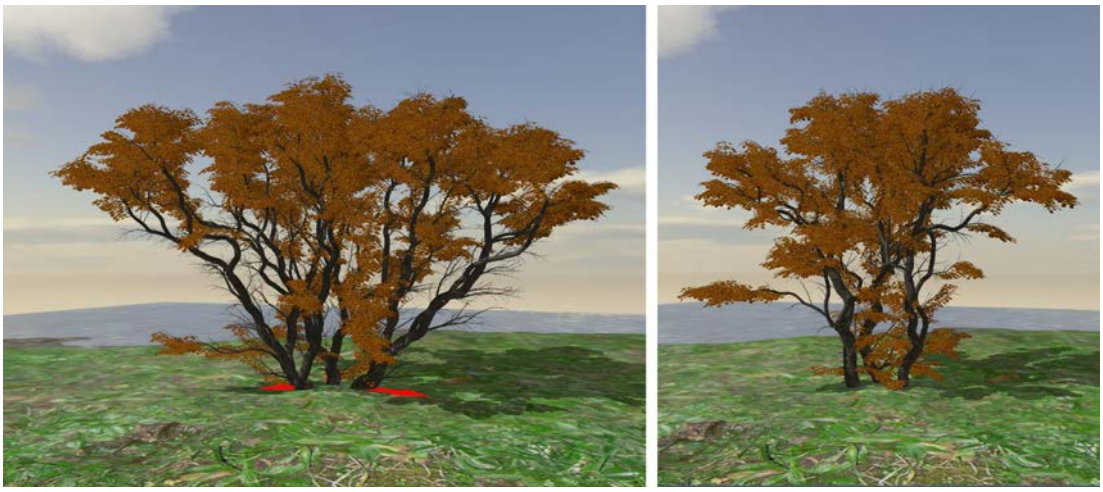


Figure 6.20: Growth with the neighborhood asymmetry tropism results in trees tilted away from their neighbors. Left) Three equally large trees tilting away from each other, neighborhood asymmetry vectors shown in red. Right) The same scene grown without neighborhood asymmetry tropism results in non-intersecting trees but without the characteristic lean seen in some real tree stands.

Figure 6.21: Larger trees have a greater effect on neighborhood asymmetry. Left) The larger tree grows unaffected while the smaller trees are tilted away (the length of red arrows represents the strength of the neighborhood asymmetry tropisms). Right) Trees do not tilt away from larger neighbors when the neighborhood asymmetry tropism is disabled.

6.7 Editing Trees

After growth, parameters of individual trees can be modified. Only the modified tree needs to be regrown while the rest of the landscape remains unchanged. The current implementation supports changing key model parameters through an interactive widget and sliders (Figure 6.22). However, a full suite of editing capabilities similar to that discussed in Chapter 5 could be implemented.

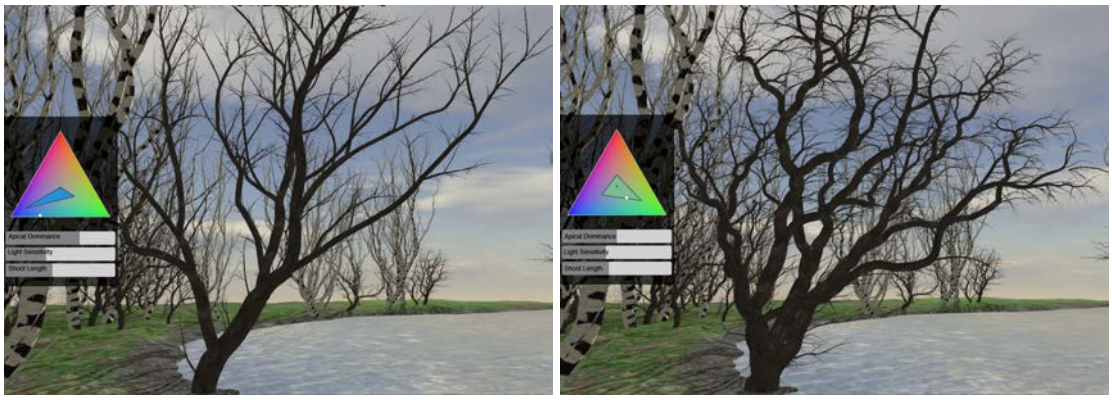


Figure 6.22: Parameters of growth can be edited individually for each tree using an interactive widget. Growth parameters for the tree on the left were edited to produce the gnarled tree on the right. Only the modified tree was regrown, all other trees remain constant.

6.8 Analysis

> "...the distribution of the crown cover centers was evidently uniformized against the contagious distribution of the stems." [Ishizuka, 1984]

The magnitude of tree crown displacement from above the tree base is a widely used measure to study crown plasticity [Ishizuka, 1984, Longuetaud et al., 2008, 2013, Schröter et al., 2012]. The above quote describes the central observation, that trees displace their

crowns in such a manner that the resulting distribution of crown centers is more uniform than the distribution of tree bases. Figure 6.23 shows an example tree base distribution overlaid with the computed tree crown centroids.

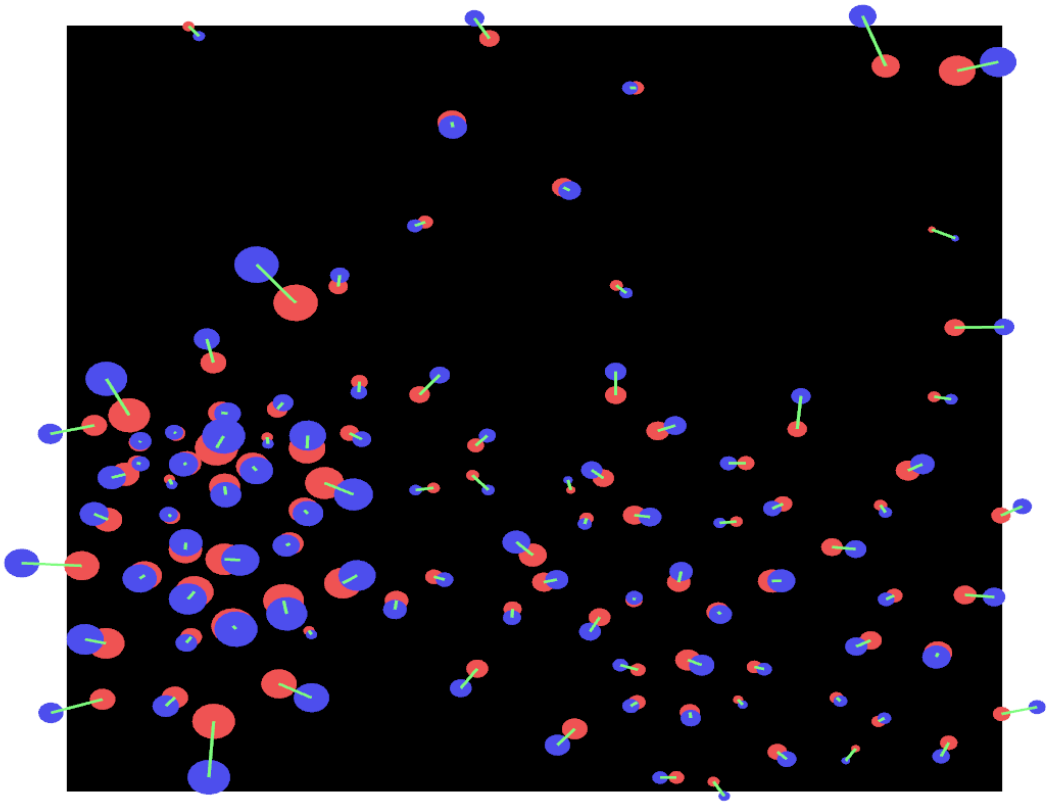


Figure 6.23: Trees growing into free space relocate the centroid of their crowns resulting in a more uniform distribution. Location of tree base is shown in red, location of tree crown centroid is shown in blue. Green lines represent correspondences. Scale of circles represents scale of tree, larger trees grow longer and thus have more potential to offset their crown. The crown centroids are relocated away from their neighboring plants.

Statistical measurements such as the Ripley-K function [Ripley, 1976] have been used to measure the uniformity of crown and base distributions [Longuetaud et al., 2008, 2013, Schröter et al., 2012]. The Ripley-K function is defined as:

$$\boldsymbol{K}(t) = \lambda^{-1} \sum_{i \neq j} \frac{I(d_{ij} < t)}{n} \tag{6.6}$$

where d_{ij} is the distance between tree i and tree j, n is the number of trees, t is the search radius ($5m$ in my measurements) and λ is the average density of trees (estimated as $\frac{n}{A}$,A being the area of the tree plot, $30m \times 30m$). $I(bool)$ is the indicator function which returns 1 or 0 if its parameter is true or false respectively. For a uniform distribution, $K(t) = \pi t^2$ [Ripley, 1976]. Therefore, the value $K(t) - \pi t^2$ can be used to measure the 'distance from uniformity'.

To validate my model, I have computed these statistics across a number of sample distributions (Figure 6.24). In all cases the distribution of crown centroids is more uniform than their bases, consistent with the results measured from real tree stands [Longuetaud et al., 2008, 2013, Schröter et al., 2012].

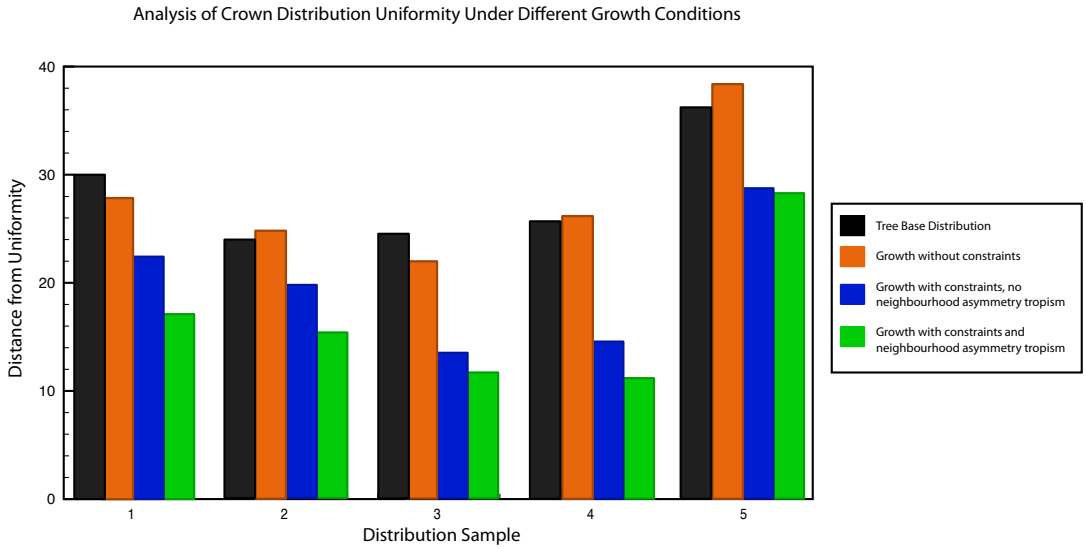


Figure 6.24: Uniformity of tree crown distribution compared to tree base distribution measured across five sample distributions (X-axis). The Y axis shows the difference between the Ripley-K number for each distribution and the Ripley-K number representing a uniform distribution. Tree base distribution is clustered and not uniform (as desired). In all cases, growth without constraints fails to significantly effect the uniformity of the crown distribution. Growth constrained by the growing discs increases the uniformity of the crown distribution. Uniformity is further increased by enabling the neighborhood asymmetry tropism. Distribution from Sample 1 is shown in Figure 6.23

6.9 Results

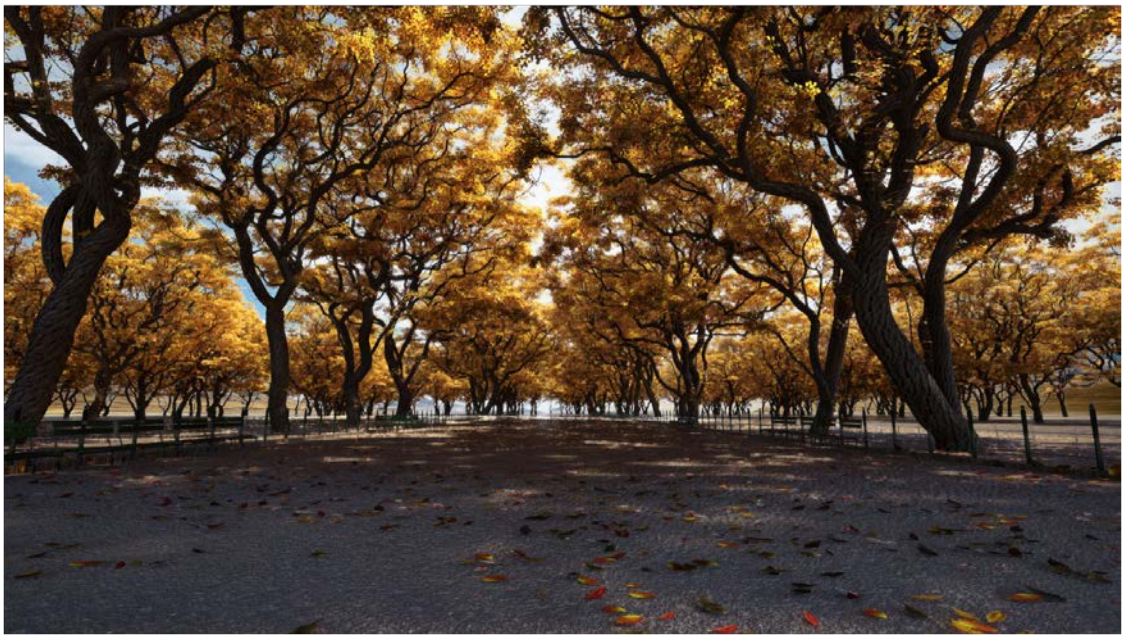


Figure 6.25: Trees over-hang a path through a city park. Scene rendered using Maya with Mental Ray.

Figure 6.26: Top) A view looking up through the canopy of the cluster of trees. Trees naturally fill the available canopy space but do not intersect. Bottom) An image looking up into a real tree canopy. (Image from http://onmaplesyrup.files.wordpress.com/2014/04/dscn6227-blog-v21.jpg)

Figure 6.27: Top) Trees at the edge of the distribution exhibit the river-bank effect. Bottom) Image of a real tree tilting away from larger neighbors. (Image from https://flic.kr/p/egGwJY)

Figure 6.28: A large tree growing around and over the top of its neighbors.

6.9.1 Speed of Simulation

The following simulations were performed on a 2011 MacBook Air, with an Intel Core i7 processor (1.8 GHz) and 4GB of RAM.

Trees	Distribution Simulation	Constraint Simulation	Growth Simulation
10	0.01s	0.17s	6.3s
30	0.02s	0.30s	12.6s
100	0.08s	0.42s	23.1s
300	0.27s	1.05s	33.9s

In all cases the plot size was constant ($30 \times 30m$). This accounts for the non-linear relation between number of trees and simulation time, as increasing the number of trees also decreases the space each one is allocated for growth and therefore the simulation time per tree is faster. The voxel space used for the constraint simulation contained $128 \times 128 \times 20$ voxels. On average, creation of tree geometry for rendering took roughly 2% of the simulation time (included in the growth simulation column). Tree growth and simulation of the constraint voxel space was split across 4 parallel threads.

Chapter 7

Conclusion

In spite of decades of computer graphics research, the modelling of three-dimensional objects has remained a tedious task. The crux of the problem is the tension between detailed control of form by the modeller and the complexity of the objects being modelled. In this thesis I have presented software and algorithms for modelling trees and landscapes which reconcile the interactive control needed in creative design with the emergence of form inherent in procedural generation. This synthesis was accomplished by integrating appropriately tailored procedural methods with a custom interface. This thesis demonstrates that:

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

My proposed algorithm for tree growth combines the realism and controllability of previous models. The diversity of generated forms has been enhanced with the incorporation of gravimorphism and tropisms quantified using the gravitropic set-point angle. The ability to produce natural looking trees of any shape is essential in the generation of landscapes.

I have extended previous landscape generation models by introducing competition between trees for crown space. The intermediate crown regions capture the effects of competition between neighboring trees and act as constraints on tree growth. This allows each tree to be simulated independently yet results in trees with the visual characteristics of those which were grown together.

TreeSketch has changed the landscape of virtual tree generation. It is the first system which makes it possible to quickly create a diverse set of natural and artistically expressive trees through an intuitive interface. A set of interactive widgets have been designed to provide intuitive control over correlated parameters. Compared to previous brush-based models [Pałubicki et al., 2009], the expressiveness of brush strokes has been significantly enhanced with the incorporation of sketch-based tropisms. Algorithms for bending branches and defining branch widths have been devised for the purpose of interactive tree modeling.

With over fifty-thousand downloads, TreeSketch users have commented saying it has changed the way they create digital trees. TreeSketch has been used by professional visual effects studios and to create trees for The Hobbit films. A wide diversity of trees can be generated with ease and delight; even children can generate realistic trees.

TreeSketch is the first software to combine interactive procedural modelling of trees with an augmented reality interface. This enables the creation and visualization of trees within real world environments, while allowing them to adapt to their surroundings. This technology opens the door to a new realm of landscaping applications allowing expensive designs to be visualized at various stages of development.

The applications that were originally envisioned for TreeSketch lie in the domain of image synthesis for computer animation, games, and computer-assisted landscape design. However, throughout our experiments with TreeSketch, we were surprised by the extent to

which it has sharpened our own perception of tree form and development in nature. Thus, possible applications of TreeSketch could be explored 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 modeller 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.

The flexible, interactive control of highly realistic procedural models achieved in this thesis defines the state of the art. It provides a benchmark for evaluating future systems for modelling trees and landscapes, and perhaps interactive 3D procedural modeling systems in general.

7.1 Future Work

A number of problems remain open for future work. I broadly categorized them below starting with those more research oriented and ending with technical advancements.

- **Interaction**: Coupling the direction of the brush strokes to the direction of branch growth has proven to be a very powerful feature, yet controlling other parameters with stroke dynamics has not been explored. Thiel et al. [2011] proposed exploiting stroke dynamics to neaten sketches by distinguishing between fine details of a stroke which are intentional, those drawn at a slow speed, from stroke noise (to be removed) in high speed sketches. These ideas could be further developed for the purpose of tree modelling, for instance, making branches closely follow the brush stroke while slowly sketching yet specify branches more generally with faster strokes.

- **Evolutionary Modelling**: Evolutionary techniques could be employed to explore the parameter space of the model and present the modeller with various versions of their tree. Different trees could be generated by simulating the growth history of the current tree (brush strokes) but modifying parameter settings in each case. Alternatively, aspects of the strokes such as speed, direction and order could be modified.

- **Landscapes**: The slope of the terrain may effect the direction of canopy displacement [Umeki, 1995]. Simulating the affect of gravity on branches while the tree is growing may reduce the need to explicitly direct away from their larger neighbors when their neighborhood is asymmetric (the neighborhood asymmetry tropism). Since the spatial constraints would already be asymmetric, the tree would naturally bend away as it grew towards free space.

- **Gravimorphism**: 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 realism of trees.

- **Branch Geometry**: Higher resolution branch geometry could be generated by a system similar to that of Mizoguchi and Miyata [2011] and textures could be procedurally generated to allow for realistic flow across branching points [Lefebvre and Neyret, 2002].

- **Animation of Plant Development**: Currently trees are only shown in their most recent stage of development. Yet, it may be desirable to show an animation of the tree developing from a seedling into its current state. Pirk et al. [2012a] have presented

a model which creates animation of tree development from static tree models. The growth model presented in this thesis already simulated the growth of a tree from the seedling. I believe this history could be leveraged to produce a superior result.

- **Augmented Reality Interface**: More advanced methods for reconstructing scene geometry would allow for a new range of plant-environment interactions including growth of climbing plants on scene geometry. To improve the visual quality of digital trees within the live video feed, illumination conditions of the outdoor environment could be captured and used to render the virtual content ([Liu et al., 2009]). Lighting conditions could also effect how plants grow in a virtual gardening application.

Bibliography

13th Lab AB. Pointcloud SDK for iOS. `http://pointcloud.io`, 2012.

M. Alsweis. Extended competition rules for interacting plants. In *Proceedings of the 15th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, pages 1–8. Václav Skala-UNION Agency, 2007.

M. Alsweis and O. Deussen. Modeling and visualization of symmetric and asymmetric plant competition. In *Proceedings of the First Eurographics Conference on Natural Phenomena*, pages 83–88. Eurographics Association, 2005.

M. Aono and T. L. Kunii. Botanical tree image generation. *IEEE Computer Graphics and Applications*, 4(5): 10–34, 1984.

D. Barthélémy and Y. Caraglio. Plant architecture: A dynamic, multilevel and comprehensive approach to plant form, structure, and ontology. *Annals of Botany*, 99:375–407, 2007.

B. Beneš, N. Andrysco, and O. Šťava. Interactive modeling of virtual ecosystems. In *Proceedings of the Fifth Eurographics Conference on Natural Phenomena*, pages 9–16. Eurographics Association, 2009.

B. Beneš, M. A. Massih, P. Jarvis, D. G. Aliaga, and C. A. Vanegas. Urban ecosystem design. In *Symposium on Interactive 3D Graphics and Games*, pages 167–174. ACM, 2011.

B. Beneš, O. Šťava, R. Měch, and G. Miller. Guided procedural modeling. *Computer Graphics Forum*, 30 (2):325–334, 2011.

J. Bloomenthal. Modeling the Mighty Maple. *Computer Graphics*, 19(3):305–311, 1985.

R. Borchert and H. Honda. Control of development in the bifurcating branch system of *Tabebuia rosea*: A computer simulation. *Botanical Gazette*, 145(2):184–195, 1984.

M. Botsch, M. Pauly, M. Gross, and L. Kobbelt. Primo: Coupled prisms for intuitive surface modeling. In *Proceedings of the Fourth Eurographics symposium on Geometry Processing*, pages 11–20, 2006.

F. Boudon, P. Prusinkiewicz, P. Federl, C. Godin, and R. Karwowski. Interactive design of bonsai tree models. *Computer Graphics Forum*, 22(3):591–599, 2003.

J. Brisson. Neighborhood competition and crown asymmetry in *acer saccharum*. *Canadian Journal of Forest Research*, 31(12):2151–2159, 2001.

X. Chen, B. Neubert, Y.-Q. Xu, O. Deussen, and S. B. Kang. Sketch-based tree modeling using Markov random field. *ACM Transactions on Graphics*, 27(5):109, 2008.

N. Chiba, S. Ohkawa, K. Muraoka, and M. Miura. Visual simulation of botanical trees based on virtual heliotropism and dormancy break. *The Journal of Visualization and Computer Animation*, 5:3–15, 1994.

E. Coen. *The art of genes: How organisms make themselves*. Oxford University Press, 1999.

D. Cohen. Computer simulation of biological pattern generation processes. *Nature*, 216:246–248, 1967.

J.-F. Côté, J.-L. Widlowski, R. Fournier, and M. Verstraete. The structural and radiative consistency of three-dimensional tree reconstructions from terrestrial LIDAR. *Remote Sensing of Environment*, 113: 1067–1081, 2009.

P. de Reffye, C. Edelin, J. Françon, M. Jaeger, and C. Puech. Plant models faithful to botanical structure and development. *Computer Graphics*, 22(4):151–158, 1988.

O. Deussen and B. Lintermann. A modelling method and user interface for creating plants. In *Proceedings of Graphics Interface*, pages 189–197, 1997.

O. Deussen, P. Hanrahan, B. Lintermann, R. Měch, M. Pharr, and P. Prusinkiewicz. Realistic modeling and rendering of plant ecosystems. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, pages 275–286. ACM, 1998.

O. Deussen, C. Colditz, M. Stamminger, and G. Drettakis. Interactive visualization of complex plant ecosystems. In *Proceedings of the Conference on Visualization'02*, pages 219–226. IEEE Computer Society, 2002.

A. Dietrich, C. Colditz, O. Deussen, and P. Slusallek. Realistic and interactive visualization of high-density plant ecosystems. In *Proceedings of the First Eurographics Conference on Natural Phenomena*, pages 73–81. Eurographics Association, 2005.

J. Digby and R. D. Firn. The gravitropic set-point angle (GSA): The identification of an important developmentally controlled variable governing plant architecture. *Plant, Cell & Environment*, 18:1434–1440, 1995.

D. Ebert, K. Musgrave, D. Peachey, K. Perlin, and S. Worley. *Texturing & modeling: A procedural approach*. Morgan Kaufmann, 2003.

E. Eisemann, M. Schwarz, U. Assarsson, and M. Wimmer. *Real-time shadows*. AK Peters, Ltd., 2011.

M. Fischler and R. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, June 1981.

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

R. Grote and H. Pretzsch. A model for individual tree development based on physiological processes. *Plant Biology*, 4(2):167–180, 2002.

L. Guibas and J. Stolfi. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Transactions on Graphics*, 4(2):74–123, April 1985.

R. W. Hall. Fast parallel thinning algorithms: parallel speed and connectivity preservation. *Communications of the ACM*, 32:124–131, 1989.

F. Hallé, R. A. A. Oldeman, and P. B. Tomlinson. *Tropical trees and forests: An architectural analysis.* Springer-Verlag, Berlin, 1978.

A. Hanson. Quaternion Gauss maps and optimal framings of curves and surfaces. *Technical Report 518, Computer Science Department, Indiana University, Bloomington, IN*, 1998.

J. Harper. *Population biology of plants.* Academic Press., 1977.

J. Harper. Modules, branches, and the capture of resources. *In Population biology and evolution of clonal organisms*, pages 1–33, 1985.

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*, 31:331–338, 1971.

B. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America*, 4(4):629–642, Apr 1987.

T. Igarashi, S. Matsuoka, and H. Tanaka. Teddy: A sketching interface for 3D freeform design. In *Proceedings of SIGGRAPH*, pages 409–416, 1999.

T. Ijiri, S. Owada, M. Okabe, and T. Igarashi. Floral diagrams and inflorescences: Interactive flower modeling using botanical structural constraints. *ACM Transactions on Graphics*, 24(3):720–726, 2005.

T. Ijiri, S. Owada, and T. Igarashi. Seamless integration of initial sketching and subsequent detail editing in flower modeling. *Computer Graphics Forum*, 25(3):617–624, 2006.

T. Ijiri, S. Owada, and T. Igarashi. The sketch L-system: Global control of tree modeling using free-form strokes. In *Proceedings of Smart Graphics*, pages 138–146, 2006.

M. Ishizuka. Spatial pattern of trees and their crowns in natural mixed forests. *Japanese Journal of Ecology*, 34:421–430, 1984.

K. Kahlen, D. Wiechers, and H. Stützel. Modelling leaf phototropism in a cucumber canopy. *Functional Plant Biology*, 35:876–884, 2008.

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

B. Lane and P. Prusinkiewicz. Generating spatial distributions for multilevel models of plant communities. In *Proceedings of Graphics Interface*, pages 69–80, 2002.

S. Lefebvre and F. Neyret. Synthesizing bark. In *Proceedings of the 13th Eurographics Workshop on Rendering*, pages 105–116. Eurographics Association, 2002.

B. Lintermann and O. Deussen. Interactive modeling of plants. *IEEE Computer Graphics and Applications*, 19(1):56–65, 1999.

M. Lipp, D. Scherzer, P. Wonka, and M. Wimmer. Interactive modeling of city layouts using layers of procedural content. *Computer Graphics Forum*, 30(2):345–354, 2011.

D. Lischinski. Incremental delaunay triangulation. *Graphics Gems IV*, pages 47–59, 1994.

Y. Liu, X. Qin, S. Xu, E. Nakamae, and Q. Peng. Light source estimation of outdoor scenes for mixed reality. *The Visual Computer*, 25(5-7):637–646, 2009.

Y. Livny, S. Pirk, Z. Cheng, F. Yan, O. Deussen, D. Cohen-Or, and B. Chen. Texture-lobes for tree modeling. *ACM Transactions on Graphics*, 30:53:1–53:10, 2011.

S. Longay, K. Kahlen, and P. Prusinkiewicz. Geometry of leaf orientation. 2010. Poster presented at Functional Structural Plant Modlelling. University of California, Davis.

S. Longay, A. Runions, F. Boudon, and P. Prusinkiewicz. Treesketch: Interactive procedural modeling of trees on a tablet. In *Proceedings of the International Symposium on Sketch-Based Interfaces and Modeling*, pages 107–120, 2012.

F. Longuetaud, T. Seifert, J-M. Leban, and H. Pretzsch. Analysis of long-term dynamics of crowns of sessile oaks at the stand level by means of spatial statistics. *Forest Ecology and Management*, 255(5):2007–2019, 2008.

F. Longuetaud, A. Piboule, H. Wernsdörfer, and C. Collet. Crown plasticity reduces inter-tree competition in a mixed broadleaved forest. *European Journal of Forest Research*, 132(4):621–634, 2013.

N. MacDonald. *Trees and networks in biological models.* J. Wiley & Sons, 1983.

W. L. Metcalf. http://www.the-athenaeum.org, All images in public domain.

G. Mc Millen and J. Mc Clendon. Leaf angle: An adaptive feature of sun and shade leaves. *Botanical Gazette*, pages 437–442, 1979.

K. Mitchell. Dynamics and simulated yield of douglas-fir. *Forest Science*, 21(Supplement 17):a0001–z0001, 1975.

A. Mizoguchi and K. Miyata. Modeling trees with rugged surfaces. In *The IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications*, pages 1464–1471, 2011.

P. Mueller, P. Wonka, S. Haegler, A. Ulmer, and L. Van Gool. Procedural modeling of buildings. In *ACM Transactions on Graphics*, volume 25, pages 614–623. ACM Press, August 2006.

K. Musgrave, C. Kolb, and R. Mace. The synthesis and rendering of eroded fractal terrains. 23(3):41–50, 1989.

C. Muth and F. A. Bazzaz. Tree canopy displacement at forest gap edges. *Canadian Journal of Forest Research*, 32(2):247–254, 2002.

C. Muth and F. A. Bazzaz. Tree canopy displacement and neighborhood interactions. *Canadian Journal of Forest Research*, 33(7):1323–1330, 2003.

R. Měch and P. Prusinkiewicz. Visual models of plants interacting with their environment. In *Proceedings of SIGGRAPH*, pages 397–410, 1996.

B. Neubert, T. Franken, and O. Deussen. Approximate image-based tree modeling using particle flows. *ACM Transactions on Graphics*, 26(3):88:1–88:8, 2007.

Ü. Niinemets and S. Fleck. Petiole mechanics, leaf inclination, morphology, and investment in support in relation to light availability in the canopy of liriodendron tulipifera. *Oecologia*, 132(1):21–33, 2002.

M. Okabe, S. Owada, and T. Igarashi. Interactive design of botanical trees using freehand sketches and example-based editing. *Computer Graphics Forum*, 24(3), 2005.

K. Onishi, N. Marukami, Y. Kitamura, and F. Kishino. Modeling of trees with interactive L-system and 3D gestures. In *Biologically Inspired Approaches to Advanced Information Technology*, pages 222–235, 2006.

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

W. Pałubicki. *A Computational Study of Tree Architecture*. PhD thesis, University of Calgary, 2012.

W. Pałubicki, K. Horel, S. Longay, A. Runions, B. Lane, R. Měch, and P. Prusinkiewicz. Self-organizing tree models for image synthesis. *ACM Transactions on Graphics*, 28(3):58:1–58:10, 2009.

S. Pirk, T. Niese, O. Deussen, and B. Neubert. Capturing and animating the morphogenesis of polygonal tree models. *ACM Transactions on Graphics*, 31(6):169:1–169:10, 2012a.

S. Pirk, O. Šťava, J. Kratt, M.A.M. Said, B. Neubert, R. Měch, B. Benes, and O. Deussen. Plastic trees: interactive self-adapting botanical tree models. *ACM Transactions on Graphics*, 31(4):50:1–50:10, 2012b.

C. Pissarro. Apple Tree at Eragny. http://www.the-athenaeum.org, All images in public domain, 1884.

J. Power, A. J. Bernheim-Brush, P. Prusinkiewicz, and D. Salesin. Interactive arrangement of botanical L-system models. In *Proceedings of the ACM Symposium on Interactive 3D Graphics*, pages 175–182, 1999.

P. Prusinkiewicz. In search of the right abstraction: the synergy between art, science, and information technology in the modelling of natural phenomena. In C. Sommerer and L. Mignonneau, editors, *Art@ Science*, pages 60–68. Springer, Wien, 1998.

P. Prusinkiewicz and A. Lindenmayer. *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.

P. Prusinkiewicz, M. James, and R. Měch. Synthetic topiary. In *Proceedings of SIGGRAPH*, pages 351–358, 1994.

P. Prusinkiewicz, M. Hammel, J. Hanan, and R. Měch. Visual models of plant development. In G. Rozenberg and A. Salomaa, editors, *Handbook of formal languages,* Vol. III*: Beyond words*, pages 535–597. Springer, Berlin, 1997.

P. Prusinkiewicz, L. Mündermann, R. Karwowski, and B. Lane. The use of positional information in the modeling of plants. In *Proceedings of SIGGRAPH*, pages 289–300, 2001.

W. T. Reeves and R. Blau. Approximate and probabilistic algorithms for shading and rendering structured particle systems. *Computer Graphics*, 19(3):313–322, 1985.

B. Ripley. The second-order analysis of stationary point processes. *Journal of Applied Probability*, pages 255–266, 1976.

Y. Rodkaew, P. Chongstitvatana, S. Siripant, and C. Lursinsap. Particle systems for plant modeling. In B.-G. Hu and M. Jaeger, editors, *Plant growth modeling and applications. Proceedings of PMA03*, pages 210–217. Tsinghua University Press and Springer, Beijing, 2003.

A. Runions, M. Fuhrer, B. Lane, P. Federl, A.-G. Rolland-Lagan, and P. Prusinkiewicz. Modeling and visualization of leaf venation patterns. *ACM Transactions on Graphics*, 24(3):702–711, 2005.

A. Runions, B. Lane, and P. Prusinkiewicz. Modeling trees with a space colonization algorithm. In *Eurographics Workshop on Natural Phenomena*, pages 63–70, 2007.

T. Sachs. Self-organization of tree form: A model for complex social systems. *Journal of Theoretical Biology*, 230:197–202, 2004.

T. Sachs. How can plants choose the most promising organs? In *Communication in Plants*, pages 53–63. Springer Berlin Heidelberg, 2006.

T. Sachs and A. Novoplansky. Tree form: Architectural models do not suffice. *Israel Journal of Plant Sciences*, 43:203–212, 1995.

T. Sachs, A. Novoplansky, and D. Cohen. Plants as competing populations of redundant organs. *Plant, Cell & Environment*, 16(7):765–770, 1993.

B. Schaudt and R. L. Drysdale. Multiplicatively weighted crystal growth voronoi diagrams. In *Proceedings of the Seventh Annual Symposium on Computational Geometry*, pages 214–223. ACM, 1991.

M. Schröter, W. Härdtle, and G. von Oheimb. Crown plasticity and neighborhood interactions of European beech (*fagus sylvatica l.*) in an old-growth forest. *European Journal of Forest Research*, 131(3):787–798, 2012.

R. Smelik, T. Tutenel, K. de Kraker, and R. Bidarra. A declarative approach to procedural modeling of virtual worlds. *Computers & Graphics*, 35(2):352–363, 2011.

A. R. Smith. Plants, fractals, and formal languages. *ACM SIGGRAPH Computer Graphics*, 18(3):1–10, 1984.

K. Sorrensen-Cothern, D. Ford, and D. Sprugel. A model of competition incorporating plasticity through modular foliage and crown development. *Ecological Monographs*, pages 277–304, 1993.

O. Šťava, S. Pirk, K. Kratt, B. Chen, R. Měch, O. Deussen, and B. Benes. Inverse procedural modelling of trees. In *Computer Graphics Forum*. Wiley Online Library, 2014.

A. Takenaka. A simulation model of tree architecture development based on growth response to local light environment. *Journal of Plant Research*, 107(3):321–330, 1994.

J. Talton, Y. Lou, S. Lesser, J. Duke, R. Měch, and V. Koltun. Metropolis procedural modeling. *ACM Transactions on Graphics*, 30(2):11:1–11:14, 2011.

P. Tan, G. Zeng, J. Wang, S.-B. Kang, and L. Quan. Image-based tree modeling. *ACM Transactions on Graphics*, 26:87:1–87:7, 2007.

Y. Thiel, K. Singh, and R. Balakrishnan. Elasticurves: Exploiting stroke dynamics and inertia for the real-time neatening of sketched 2d curves. In *ACM Symposium on User Interface Software and Technology*, pages 383–392, 2011.

S. Ulam. On some mathematical properties connected with patterns of growth of figures. *Proceedings of Symposia on Applied Mathematics*, 14:215–224, 1962.

K. Umeki. A comparison of crown asymmetry between *picea abies* and *betula maximowicziana*. *Canadian Journal of Forest Research*, 25(11):1876–1880, 1995.

H. M. Ward. *Trees. Volume V: Form and habit*. Cambridge University Press, Cambridge, 1909.

M. Ward. Xmdvtool: Integrating multiple methods for visualizing multivariate data. In *Proceedings of the Conference on Visualization'94*, pages 326–333. IEEE Computer Society Press, 1994.

J. Weber and J. Penn. Creation and rendering of realistic trees. In *Proceedings of SIGGRAPH*, pages 119–128, 1995.

C. F. H. Werner. Abraham's Tree Near Hebron. http://www.the-athenaeum.org, All images in public domain, 1862.

J. White. The plant as a metapopulation. *Annual Review of Ecology and Systematics*, 10(1):109–145, 1979.

J. Wither, F. Boudon, M.-P. Cani, and C. Godin. Structure from silhouettes: A new paradigm for fast sketch-based design of trees. *Computer Graphics Forum*, 28(2):541–550, 2009.

L. Xu and D. Mould. A procedural method for irregular tree models. *Computers & Graphics*, 36(8): 1036–1047, 2012.

M. Zakaria and S. Shukri. A sketch-and-spray interface for modeling trees. In *Proceedings of Smart Graphics*, pages 23–35, 2007.

Appendix A

Bending Branches

This chapter presents an adaptation of the mesh deformation method PriMo [Botsch et al., 2006] to tree skeletons. PriMo represents mesh geometry as a set of rigid prisms connected by elastic joints. The modeller 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 TreeSketch, prisms correspond to the internodes in the path between a modeller-selected base internode B and end internode E. Each prism is represented by its eight vertices. The faces separating a pair of adjacent prisms, P_i and P_j, are noted $\boldsymbol{f}^{i\to j}$ and $\boldsymbol{f}^{j\to i}$. They are represented by bilinear patches, with the u,v axes aligned with the $\boldsymbol{U}$p and $\boldsymbol{L}$eft axes of the $\boldsymbol{HLU}$ coordinate frame of their corresponding internode. Between these patches resides an elastic joint connecting P_i and P_j (Figure A.1).

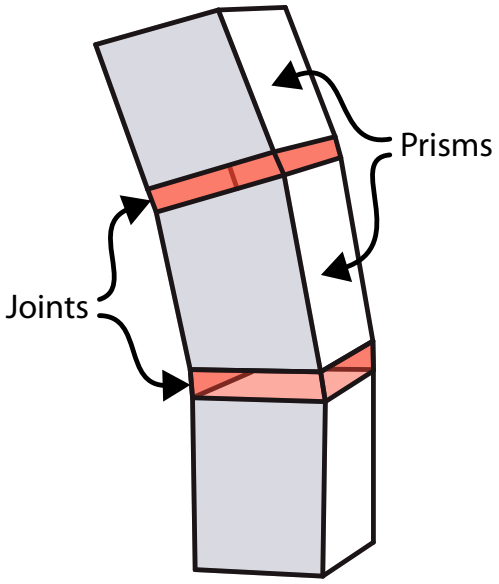


Figure A.1: Representation of branch geometry for the PriMo deformation method. Prisms correspond to internodes, joints correspond to nodes of a path in a tree. The prisms have been stretched apart for illustration, there are no gaps between prisms in the initial state of the system. Figure from Longay et al. [2012].

Botsch et al. [2006] defines the energy between adjacent prisms P_i and P_j as the integral of the squared distances between corresponding points on the faces $f^{i\to j}$ and $f^{j\to i}$:

$$E_{ij} = \int_{[0,1]^2} \|\boldsymbol{f}^{i\to j}(u,v) - \boldsymbol{f}^{j\to i}(u,v)\|^2 du dv \tag{A.1}$$

This function approximates the energy incurred by the stretching of 'fibers' at the elastic joint connecting the two prisms. To minimize this energy, I solve for the optimal rotation $\boldsymbol{R}_i$ and translation $\boldsymbol{t}_i$ of each unconstrained prism, which minimizes the weighted sum of energy functions E_{ij}, yielding the following equation [Botsch et al., 2006]:

$$\min_{\boldsymbol{R}_i,\boldsymbol{t}_i} \sum_{j\in\mathcal{N}_i} w_j \int_{[0,1]^2} \|\boldsymbol{R}_i \boldsymbol{f}^{i\to j}(u,v) + \boldsymbol{t}_i - \boldsymbol{f}^{j\to i}(u,v)\|^2 du dv \tag{A.2}$$

Here $\mathcal{N}_i$ represents the set of prisms adjacent to P_i and w_j is a weight associated with P_j. Inspired by Power et al. [1999], I assign a weight to each prism corresponding to its flexural stiffness. The flexural stiffness of a branch, or its resistance to bending, is defined as the product of its elastic modulus and the second moment of area. Assuming that all branches in the tree are made of the same material, the elastic modulus and constants can be safely ignored. The weight is thus defined as:

$$w_i = \frac{r_i^4}{l_i} \tag{A.3}$$

where r_i and l_i are the radius and the length of the internode respectively. This weight results in a more natural feel when pulling on branches as thicker branches deform relatively less than thin branches.

The position and orientation of each prism are computed iteratively for each prism using the method proposed by Horn [1987]. This technique only considers the neighboring prisms and thus is referred to as local shape matching. The system of Botsch et al. [2006] was designed to deform large mesh structures with many prisms. Therefore, they require a hierarchical shape matching technique in which many iterations of global matching are followed by this local shape matching. However, branches tend to have a relatively small number of elements and I found it sufficient to use only local shape matching.

By construction, the global minimum of Equation A.2 is the initial undeformed state. This facilitates circular editing [Lipp et al., 2011] as moving the selected prisms back to their initial location undoes the deformation. As noted by Botsch et al. [2006], scaling the

radius of the prism geometry provides control over the susceptibility of joints to stretching and bending (Figure 5.16). Although real tree branches are almost non-stretchable, I found it convenient to allow for stretching when editing branches. This is particularly useful to elongate a trunk.

The tree is manipulated by changing the position of the selected internodes with a multi-touch gesture (Figure 5.17). 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. As an axis is deformed, other branches maintain their relative positions and orientations with respect to their supporting internodes. Bending branches can be used to create a variety of novel tree forms as shown in Figure 5.18.

Appendix B

Depth Constraints

When investigating artistic depictions of trees it was noticed that crossing of large branches tended to be avoided (Figure B.1). In the case where large branches do cross, they cross at nearly right angles. Yet, when taking photographs of trees it is extremely difficult do obtain such a view. Moreover, photographs of trees are often denser than their artistic counterparts. Thus, it seems many artists choose to create trees which look visually appealing but are not sufficiently dense to represent a 3D structure. While not biologically accurate, this technique produces visually pleasing images.

Figure B.1: Paintings of trees. Large branches are only shown crossing at near right angles. If these trees were reconstructed in 3D their structure would appear sparse. Left: [Werner, 1862], Right: [Pissarro, 1884]

To create these forms, I introduce the concept of depth constraints, where the set of attraction points is constrained to lie within a modeller controlled depth from a plane centered at the base of the tree. All points that reside outside this depth are removed before affecting growth (Figure B.2). I explored pruning those branches which extended beyond the specified depth as in [Prusinkiewicz et al., 1994], but this resulted in a harsh, dense structure at the boundary. By restricting locations of attraction points, the tree may extend slightly outside the constrained depth, but the result is a more natural structure. Conveniently, trees created using a depth constraint have considerably less geometry than their fully 3D counterparts which is beneficial for many graphics applications which only require a single view of the tree.

Figure B.2: Examples of trees grown with a constrained depth. The white plane at the base of the trees represents the size of the depth constraint.

Appendix C

Details of Tree Appearance

While the form of the tree is mostly conveyed through the path of its branches, many small details are necessary to produce realistic results. In this chapter I discuss the placement and orientation of leaves and organs, and the geometry used to represent branches.

C.1 Leaves

Algorithms for modelling trees often focus on creating realistic branch architectures and do not explicitly address the question of leaf orientation. In the following section I discuss models for leaf placement and reorientation, which increase the visual appearance of tree models without relying on a detailed computation of the light environment for individual leaves.

C.1.1 Placement

The visual appearance of a tree is greatly effected by the density and size of leaves (Figure C.1). In real trees, a leaf is positioned at the base of each new lateral bud. However, in an interactive application, constraining the location of leaves to the location of buds is not desirable as the modeller would need to regenerate the tree with a shorter internode length to get denser leaves. Therefore, to provide more artistic freedom these two properties are detached and leaves are positioned on all branches with a width below a modeller-defined threshold.

Figure C.1: The effects of leaf density on tree form. Left) Sparse large drooping leaves produce the feeling of a small structure. Center) Small dense leaves cause the structure to take on a grander character. Right) Changing the left color and texture dramatically effects its perception.

C.1.2 Orientation

The initial orientation of leaves is defined by a phyllotactic pattern. Through responses to their environment, leaves are able to assume a more favorable position [Kahlen et al., 2008]. Some plants, primarily trees that are often exposed to the full radiation of the sun, orient their leaves in a direction that is more parallel to the sun's rays [Niinemets and Fleck, 2002]. This helps to control leaf temperature and, as more light propagates onto the crown, it also enhances the radiation received by the lower layers of foliage, thus increasing the photosynthetic ability of the tree as a whole [Millen and Clendon, 1979]. On the other hand, under-story plants, which have limited light exposure, tend to orient their leaves as horizontally as possible [Niinemets and Fleck, 2002].

Mechanisms of Leaf Orientation

Leaf movement in either the horizontal or vertical directions can be controlled through a specialized motor organ, the pulvinus, or through differential changes in cell volume on

opposite sides of the petiole. In addition, although the mechanics are less evident, twisting along the petiole can also occur. For instance, in a pea plant, regardless of the orientation of the stem, flowers produced will always be oriented in an upwards direction. At times, the petiole is required to perform a twist of 180° to obtain the desired orientation [Coen, 1999].

Model

A petiole is represented as an inextensible rod of length S, parameterized by the arc-length distance from the point of petiole attachment to the branch to a given point $P(s)$. Each point $P(s)$ of the petiole is associated with a local frame of reference defined by mutually orthogonal unit vectors $\boldsymbol{H}$, $\boldsymbol{L}$ and $\boldsymbol{U}$ (heading, left and up). Assuming that the vector $\boldsymbol{H}$ is tangent to the petiole, and the vectors $\boldsymbol{L}$ and $\boldsymbol{U}$ are aligned with the principal axes of the petiole cross-section. At the junction between the petiole and leaf blade, these vectors are parallel and perpendicular to the leaf blade, respectively (Figure C.2).

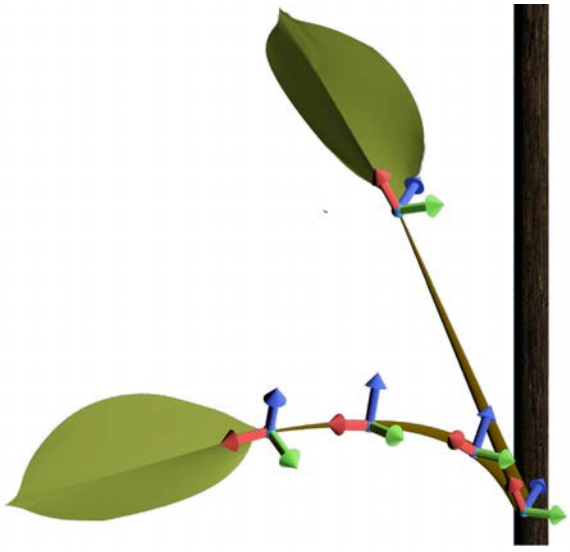


Figure C.2: The result of leaf orientation on the petiole. The frame associated with the leaf petiole, H (Red) L (Green) U (Blue).

In general, two successive frames of reference separated by an infinitesimal rod segment of length ds are rotated with respect to each other. This infinitesimal rotation is represented as a vector $d\phi$. The vector $\boldsymbol{\Omega} = \frac{d\phi}{ds}$ calculated at point $P(s)$ is the rate of petiole reorientation at $P(s)$. Considering the modes of leaf movement, the rate of rotation $\boldsymbol{\Omega}$ is decomposed into components: turning around the $\boldsymbol{U}$ axis, $\Omega_U = \boldsymbol{\Omega} \cdot \boldsymbol{U}$, bending around the $\boldsymbol{L}$ axis, $\Omega_L = \boldsymbol{\Omega} \cdot \boldsymbol{L}$, and twisting around the $\boldsymbol{H}$ axis, $\Omega_H = \boldsymbol{\Omega} \cdot \boldsymbol{H}$. These component rates of rotation are calculated as functions of the difference between current and target orientation. Considering a rotation around vector $\boldsymbol{H}$ aimed at aligning vector $\boldsymbol{U}$ with a given tropism vector $\boldsymbol{T}$. If $\boldsymbol{T}$ points upward, this rotation will twist the petiole such that the leaf blade will be brought closer to a horizontal position. The rate of twist at point $P(s)$ is modelled as if $\boldsymbol{T}$ was a force acting on the arm $\boldsymbol{U}$, creating a torque $\boldsymbol{M} = \boldsymbol{U} \times \boldsymbol{T}$. The component of this torque acting along the axis $\boldsymbol{H}$ has the magnitude $\boldsymbol{H} \cdot \boldsymbol{M} = \boldsymbol{H} \cdot (\boldsymbol{U} \times \boldsymbol{T}) = (\boldsymbol{H} \times \boldsymbol{U}) \cdot \boldsymbol{T} = -\boldsymbol{L} \cdot \boldsymbol{T}$. The rate of rotation Ω_H is assumed to be proportional to this moment, $\Omega_H = -e_H \boldsymbol{L} \cdot \boldsymbol{T}$, where e_H is a parameter characterizing the susceptibility of the petiole to twisting. Analogous reasoning leads to the formulae that capture other rotations, $\Omega_L = e_L \boldsymbol{H} \cdot \boldsymbol{T}$ and $\Omega_U = e_U \boldsymbol{L} \cdot (\boldsymbol{T} \times (\boldsymbol{H} \times \boldsymbol{T}))$.

For computational purposes, a petiole is approximated as a sequence of segments of length Δs. Composing the component rotations described above specifies reorientation at each node. The reorientation process then progresses through the petiole in the proximal to distal order, defining the shape of the petiole and the position and orientation of the leaf at its end. Figure C.3 shows the results of leaf orientation on a tree structure.

Figure C.3: Leaf orientation

C.2 Fruit

A model for determining whether or not a bud should flower was proposed by Pałubicki [2012]. In this model, if the vigor of a bud is above a threshold it will produce a flower. Since flowers produce fruit if fertilized, a similar technique is utilized here to compute the placement of fruit within the tree crown. Instead of using the vigor of a bud to determine its fate, the light exposure of the bud is used directly. This provides more intuitive control over the fruit density since light exposure is not parameterized while the vigor of the bud is effected by many parameters such as gravimorphism and apical dominance.

Only buds that reside on thin branches of the tree are considered for fruit placement. To decide which of these buds will be fruit-bearing, a probability is assigned to each bud using the following equation:

$$p = rL^2 \qquad 0 \leq L, r \leq 1 \tag{C.1}$$

L represents the light exposure of the bud (Chapter 4) and the exponent helps to accentuate its effect. r is a pseudo-random number computed for each bud by selecting from a precomputed table of random numbers using a position-dependent hash function. The table index is computed using the bitwise-XOR function to combine the components of the 3D bud position. All buds that have a probability above a modeller-defined threshold will produce fruit at their location. Fruit models are oriented in the frame of their supporting bud with a modeller-defined parameter *Droop* used to rotate the frame downwards. Figure C.4 shows the more natural effect obtained from using light exposure to control probability instead of using a purely random probability.

Figure C.4: Comparison of flower placement methods. Left) Buds with more light exposure have a higher probability of bearing a fruit. Right) Similar fruit density where probability of bearing a fruit is random.

C.3 Branches

Branch geometry is represented using generalized cylinders, with each cylinder segment corresponding to an internode. A bark texture is tiled along the path of branches and slightly rotated at each segment to avoid repetitive artifacts. To reduce polygon count on thin branches, the number of contour sides is dynamically adjusted and determined by the width at the base of each branch (Figure C.5). The thinnest branches in the tree, of width d_e, are represented with 3 sided cylinders. For thicker branches of width d, the number of sides C is computed with the following formula:

$$C = \frac{3d^n}{d_e} \tag{C.2}$$

When $n = 1$ the circumference represented by each face is constant. The parameter $n > 1$ allows thicker branches to be represented by less polygons.

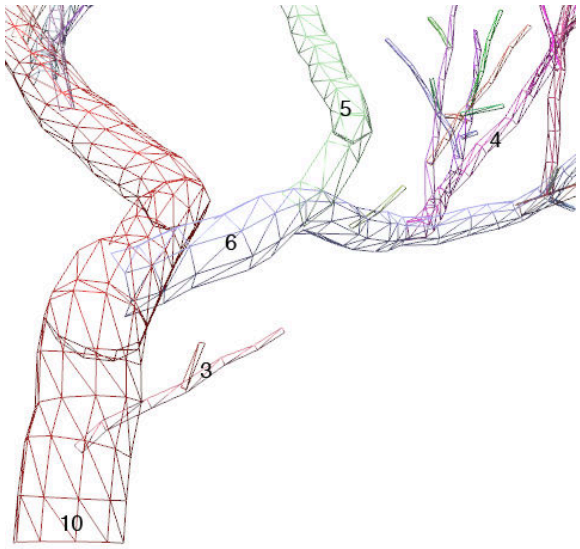


Figure C.5: Figure showing the dynamic change in contour sides determined by the width of branches. Labels represent the number of contour sides used to represent the underlying branch.

The tree generating algorithm proposed in Chapter 4 associates branches with the growth path of a single apex. When a lateral bud grows, it creates a new branch which becomes a child of its supporting axis. This branch topology, however, does not always correspond to the visual topology of the tree structure. Visually, the topology of a tree structure can be defined by recursively associating a branch to the thickest path from its base to a tip. All branches attached become its children. Figure C.6 shows a branching structure where these two definitions of branch topology are not equivalent.

Figure C.6: An example where the thickest path is actually composed of many orders of branching. (Image from http://home.cc.umanitoba.ca/ remphre/crooked.shtml)

If the cylinders representing branch geometry are created with the topology defined by the path of each apex, cracks appear at branching points where lateral branches dominate their parent axis (Figure C.7). Therefore, when creating branch geometry, I reorganize branching structure following the thickest path topology, greatly increasing the resulting mesh quality.

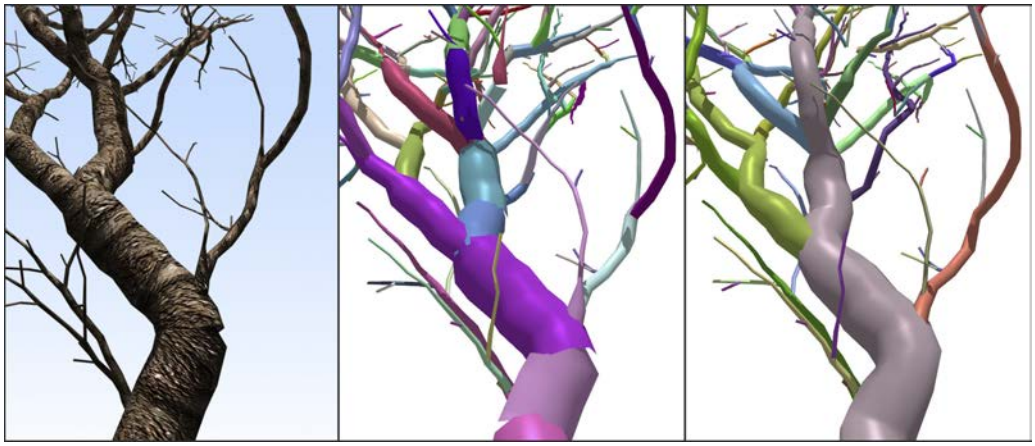


Figure C.7: Cracks in branch geometry. Left) Textured tree with cracks. Center) Colors show the topological structure of the tree. Cracks appear when a lateral branch dominates its parent axis. Right) Cracks in geometry are fixed using the proposed thickest path topology.

Appendix D

Modelling a Tree with TreeSketch

This chapter presents a sample tree design work flow. TreeSketch allows the modeller to create trees by freely interweaving different modes of operation. This work flow shows one possible method to create a tree, however, the modeller is not restricted to these steps. In this case, the modeller uses a reference image to motivate the design process (Figure D.1).

Figure D.1: Photograph of a real birch tree used as a reference image to model the tree in Figure D.2. Image from: https///flic.kr/p/jHcH

Starting from a seedling, the paths of main branches are specified by sketching (Figure D.2 A). Additional branches, filling out the crown, are then introduced using the autonomous growth (Figure D.2 B).

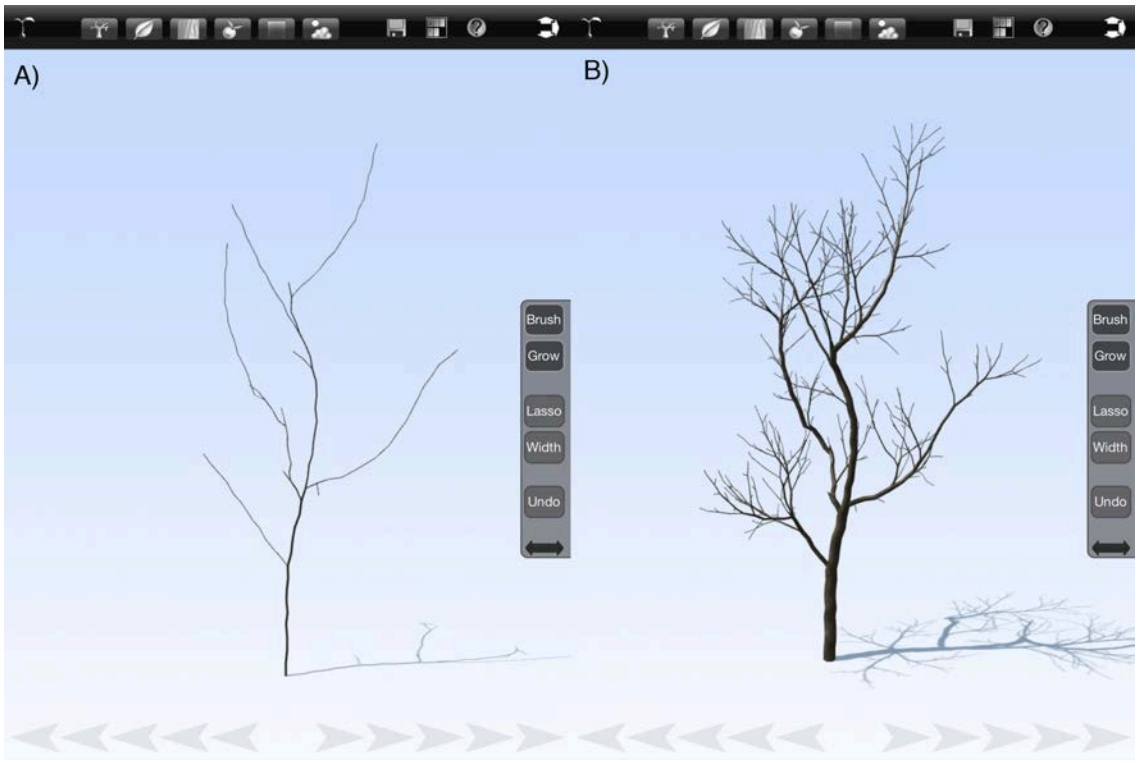


Figure D.2: The process of creating a birch tree. Details explained in the text.

Next, the modeller changes the growth parameters to produce long downwards shoots and grows the structure further (Figure D.3 C). Finally, the modeller fills out the base of the tree by brushing, and adjusts the width of branches and bark texture to better match the reference image (Figure D.3 D).

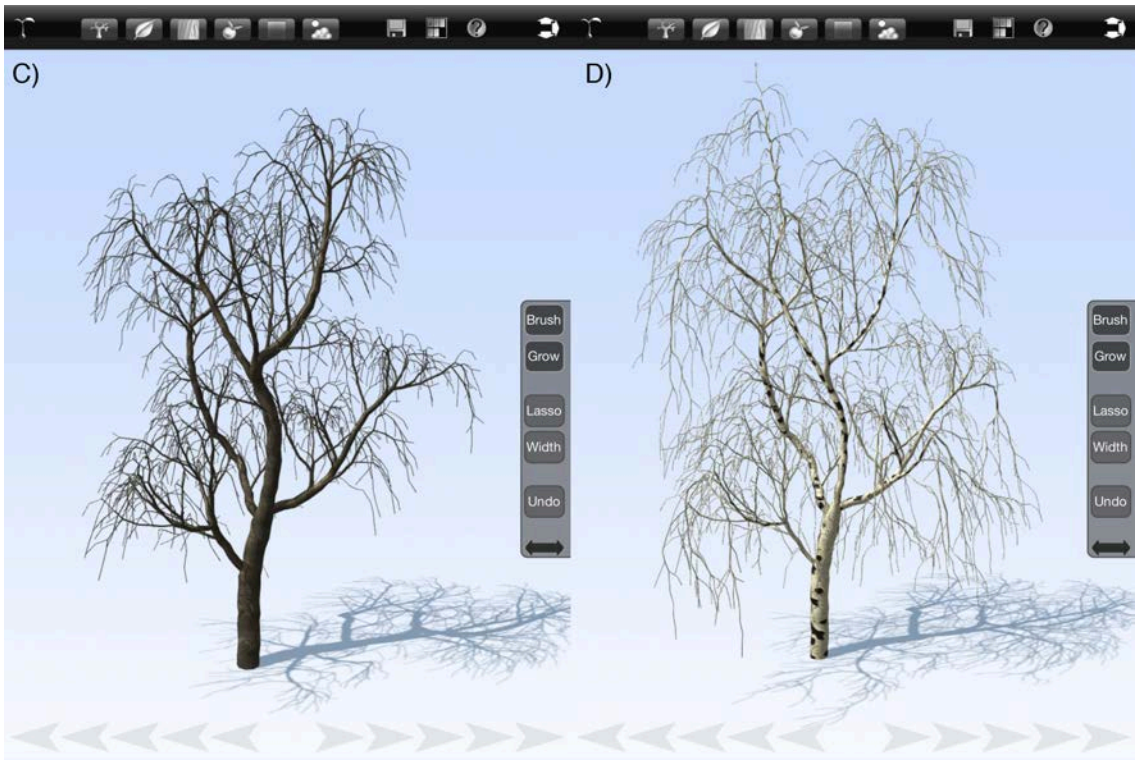


Figure D.3: The process of creating a birch tree. Details explained in the text.

While the resulting tree does not exactly match the reference photograph, it faithfully captures its character. A closer match could be obtained by bending individual branches. At this stage the tree can be exported for incorporation into a scene.

A detailed instruction manual explaining the interface and parameters is distributed with TreeSketch. Video tutorials have also been created to explain process of creating a tree and demo key features (https://vimeo.com/user7899797/videos).

Appendix E

Interface Panels of TreeSketch

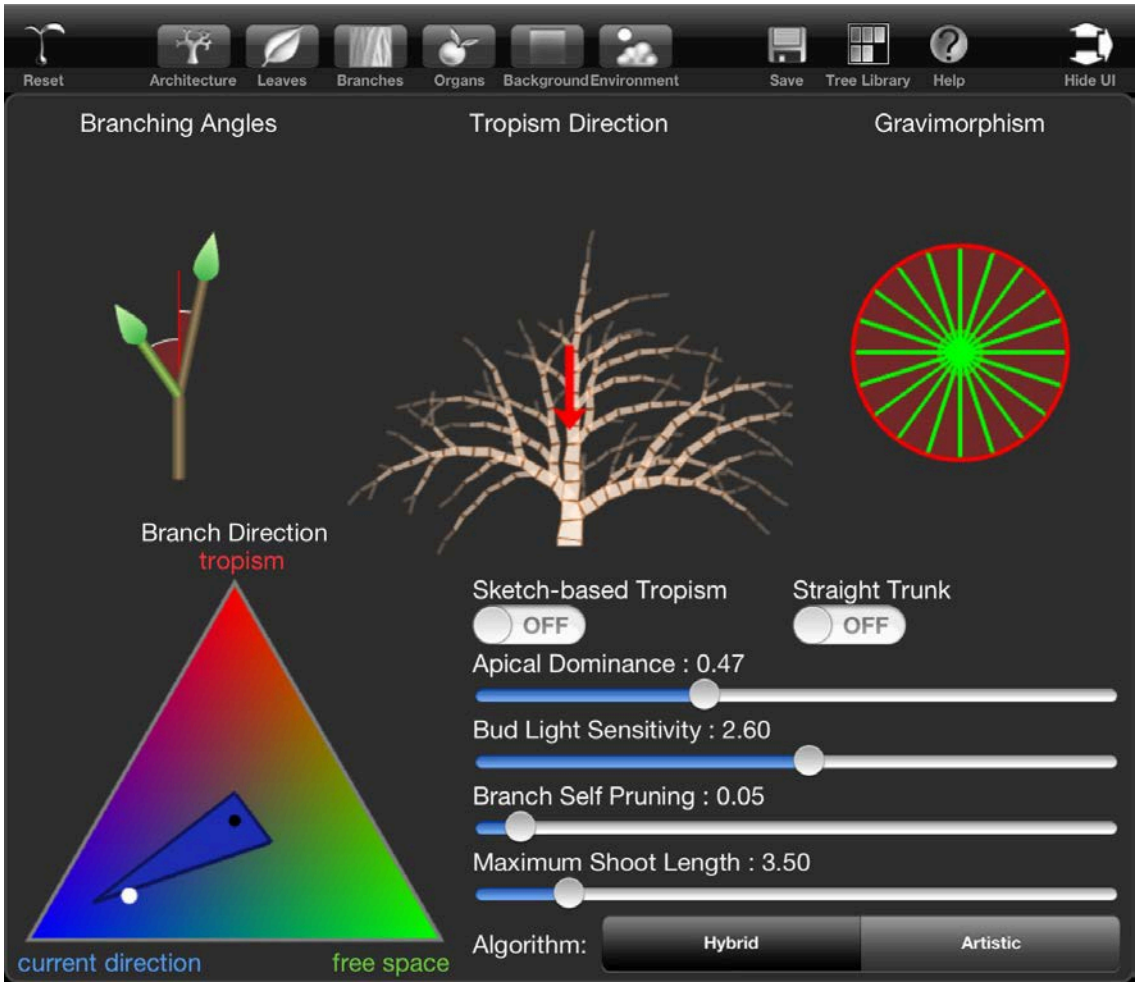


Figure E.1: TreeSketch architectural panel

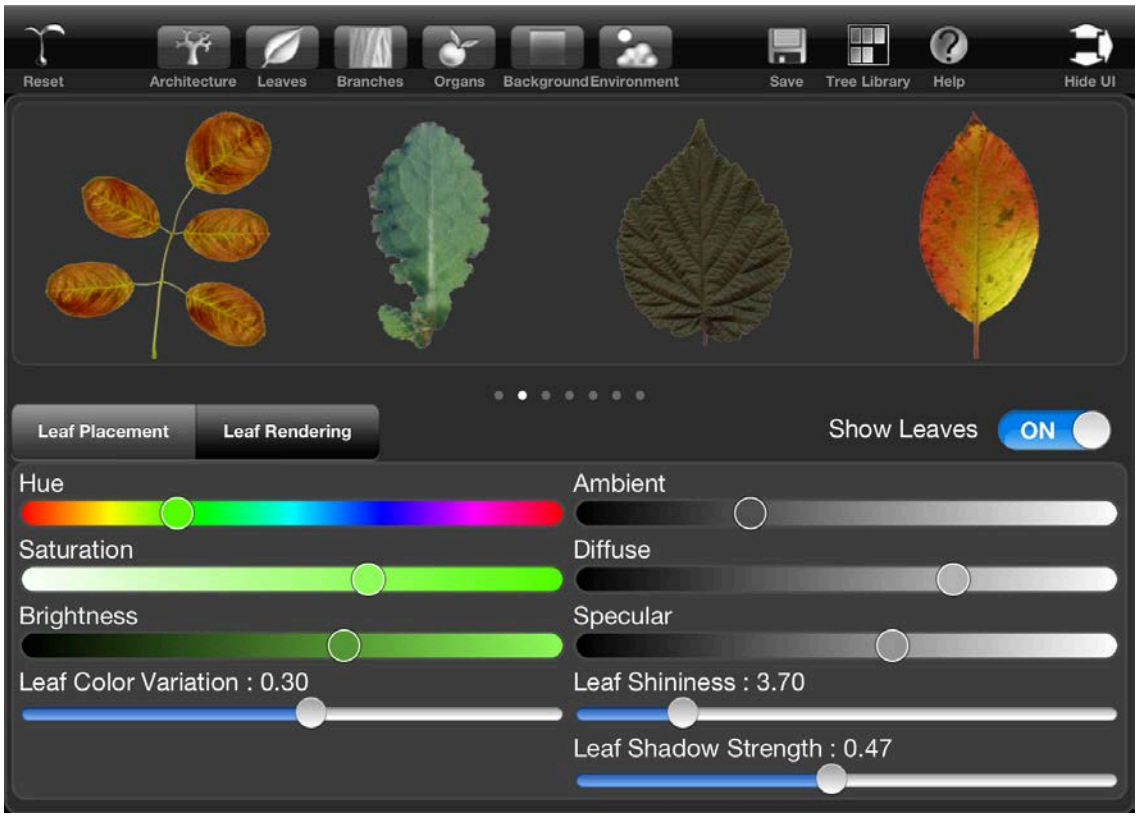


Figure E.2: TreeSketch leaves panel with showing the rendering tab.

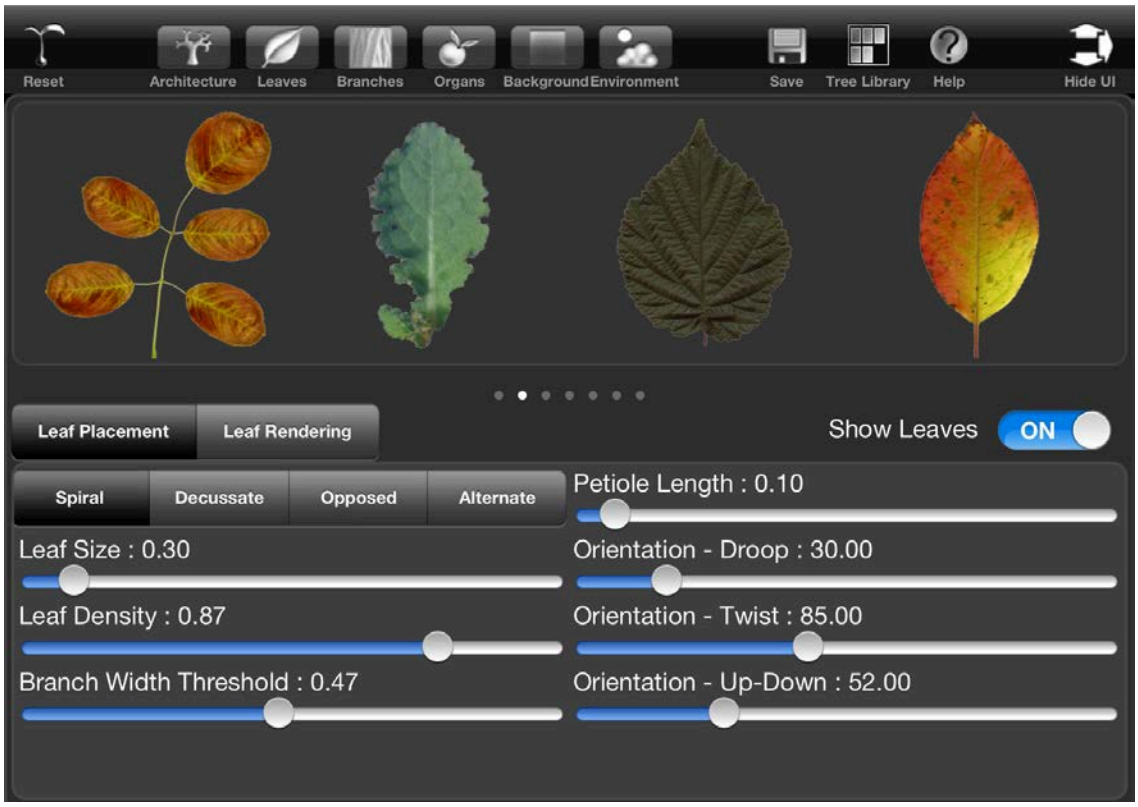


Figure E.3: TreeSketch leaves panel with showing the placement tab.

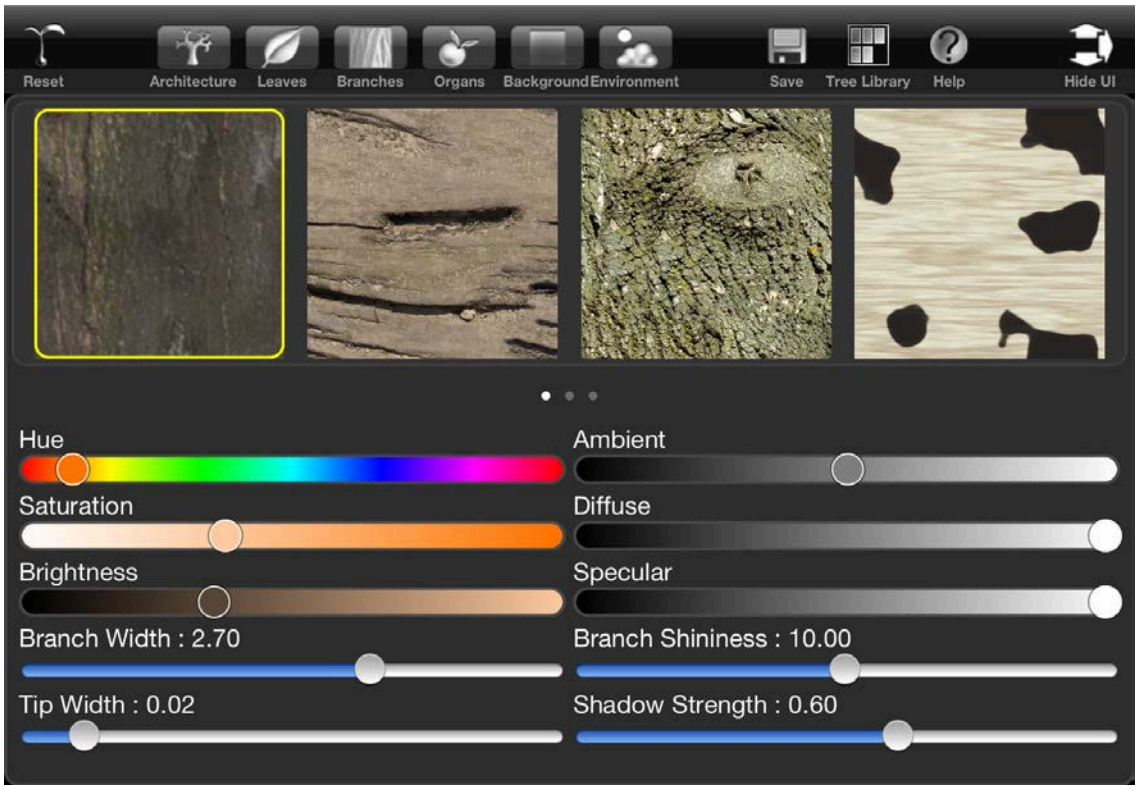


Figure E.4: TreeSketch branches panel

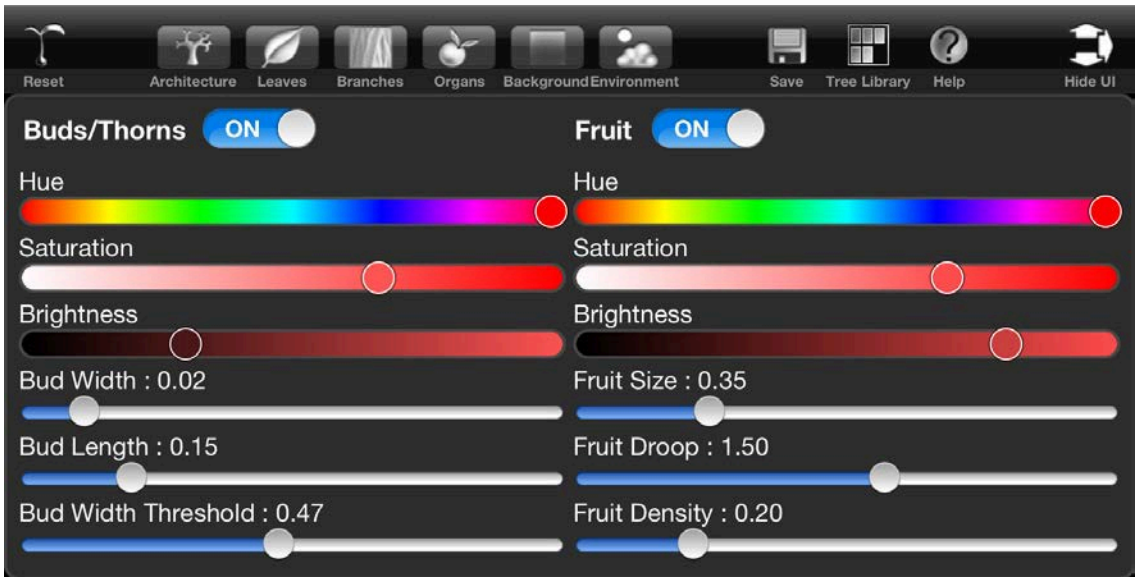


Figure E.5: TreeSketch fruit and buds panel

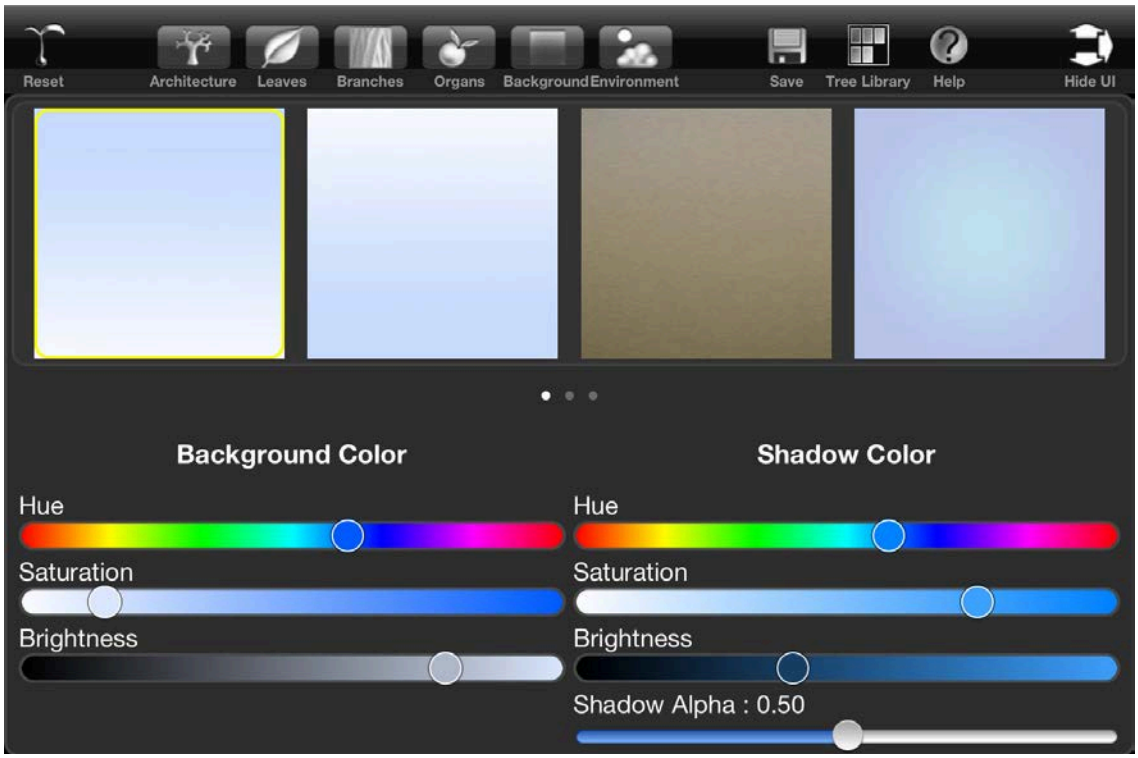


Figure E.6: TreeSketch background selection panel

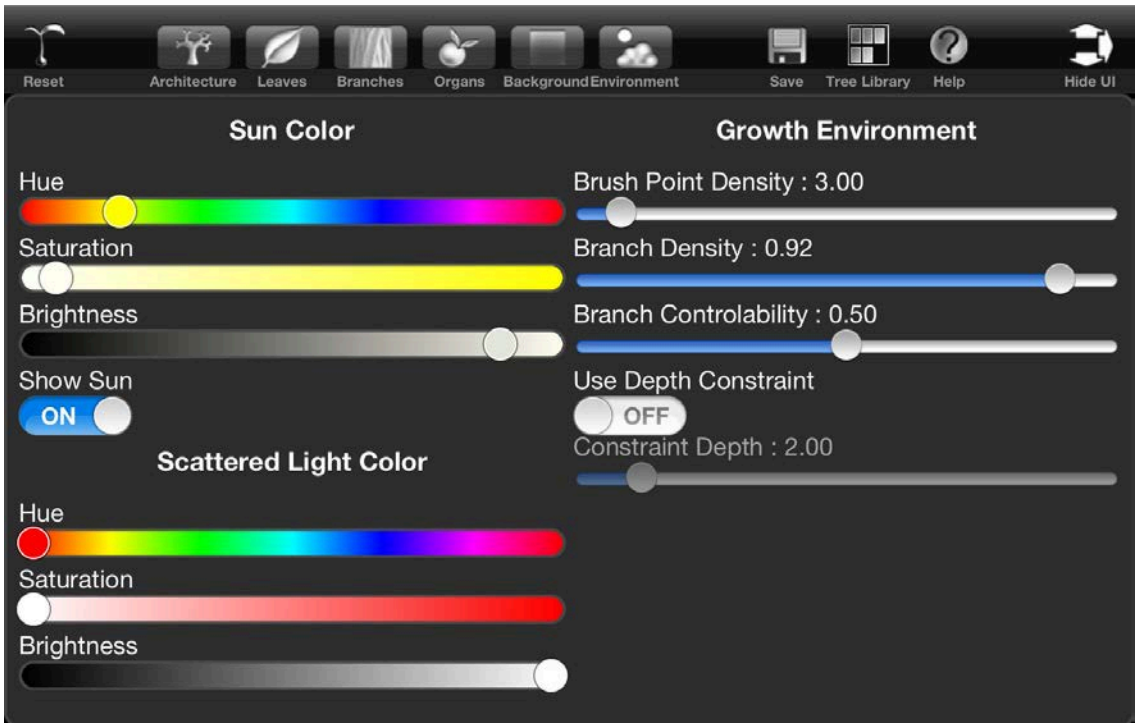


Figure E.7: TreeSketch environment panel

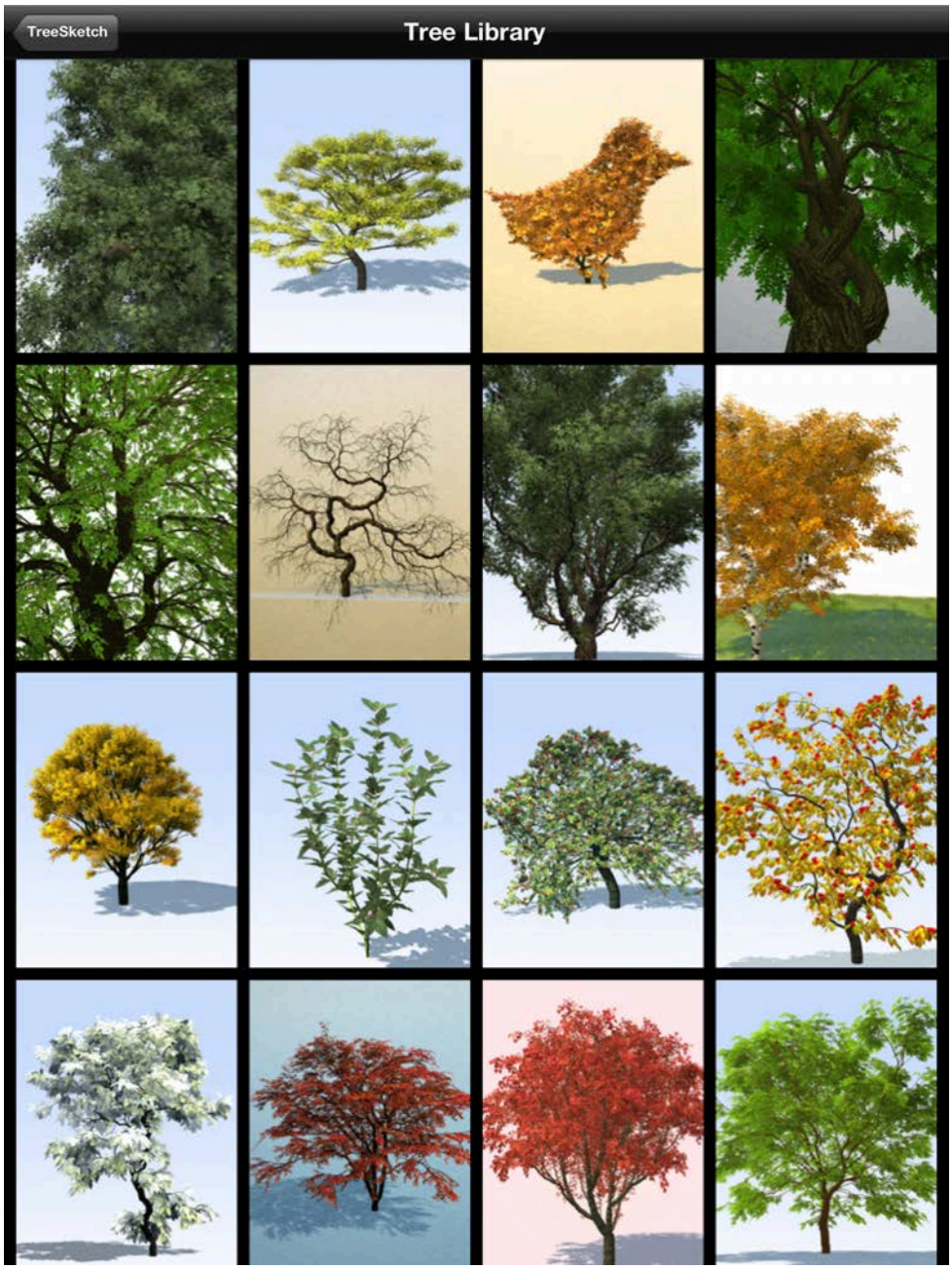


Figure E.8: The tree library where saved trees can be viewed, reloaded for further manipulation and exported. The parameters of saved trees can also be loaded, acting as growth or rendering presets