THE UNIVERSITY OF CALGARY

Application of implicit methods to the interactive modeling of trees

by

Vishal Kochhar

A THESIS
SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE
DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

CALGARY, ALBERTA

January, 2010

Abstract

Recently, Runions et al. proposed a method for modeling trees, known as the space colonization algorithm (SCA) [83]. The SCA works by distributing points in the space enclosed by a surface (crown envelope). A tree grows towards these points and consumes them in the process. In this thesis, implicit methods are used for extending the range of trees that can be created by the SCA. Point distributions are central to the SCA, and implicit surfaces are naturally suited for generating these distributions, since it is easy to determine if a point belongs inside or outside an implicit surface. Furthermore, the field values of implicit surfaces can also be used to control the density of point distributions.

An implicit modeling system was developed for the interactive specification of crowns, and for extending the range of crown shapes that can be defined, thus extending the flexibility of the SCA. This flexibility was further increased by generating different point distributions. A particularly interesting case was to limit point sets outside and in proximity to surfaces, thus causing trees to grow around surfaces, and allowing shapes to be suggested by the surrounding branching structures.

Acknowledgements

Every work requires constant support and guidance from many others, and this thesis is no exception. First and foremost, I would like to express my gratitude to Dr. Przemyslaw Prusinkiewicz, my supervisor. I consider myself very fortunate to have learned many invaluable things from him. Dr. P. has provided insightful comments and ideas throughout my research that have tremendously helped me with the writing of this thesis. Thanks Dr. P. for your continued support, funding, revisions to the thesis, guidance, and priceless ideas.

I am indebted to Dr. Brian Wyvill, my co-supervisor, for introducing to me the wonderful and fascinating world of computer graphics, for constant encouragement, and funding. His ideas had have a profound effect on my learning. I owe a big thanks to Dr. Faramarz Samavati. The implicit modeling system started as a project for his outstanding modeling course. Thanks to my examination committee for their time, their feedback on this thesis, and for making the defense process very enjoyable.

Then there are the members of the Jungle lab, who have provided such a stimulating working environment. Thanks to Adam Runions, Brendan Lane, and Erwin de Groot for a lot of invaluable advice. My thanks also goes to Jey Chelladurai, Wojtek Palubicki, Richard Smith, Pierre Barbier de Reulle, John Brosz, Luke Olsen, and Steve Longay for their help on innumerable occasions.

Finally, these acknowledgements would be incomplete without mentioning my family, and most especially my parents, my wife, and my son Neel, who have provided constant support and encouragement, and have truly blessed my life.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Recently, Runions et al. proposed a method for modeling trees, known as the space colonization algorithm (SCA) [83]. The SCA works by distributing points in the space enclosed by a surface (crown envelope). A tree grows towards these points and consumes them in the process (Figure 1.1). In this thesis, implicit methods are used for extending the range of trees that can be created by the SCA.

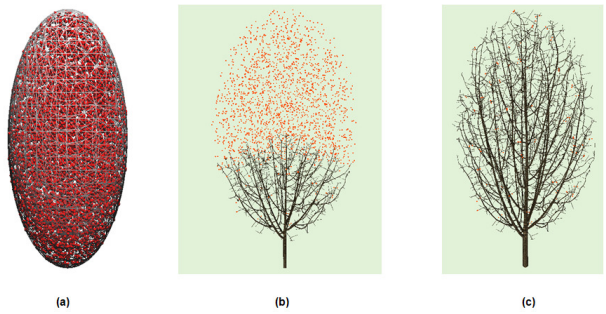


Figure 1.1: The Space colonization algorithm. a) A crown envelope populated with points; b) A tree growing toward the points; c) The generated tree.

In the SCA, there are three ways by which a tree form can be controlled — by defining the crown envelope, by varying the point distributions, and by controlling the parameters of the SCA. In the original SCA, the emphasis was on the third step, while the first two steps were relatively underplayed.

In this thesis, the flexibility of the SCA is increased by concentrating on the first two steps. Specifically, implicit methods are used for extending the range of crown shapes that can be defined, and for generating different point distributions in the

space enlclosed by these crowns. The property of implicit surfaces that make them particularly attractive for these tasks is that it is very easy to determine if a point belongs inside or outside an implicit surface. This allows for the extension of crown envelopes, since they can be easily populated with points. In contrast, parametric or subdivision surfaces can also be used for defining arbitrary crowns, but since these crowns can't be easily populated with points, they are more difficult to apply to the SCA. Another attractive property of implicit surfaces is that the density of point distributions within a region can be controlled using the field values of implicit surfaces.

An implicit modeling system was developed for the interactive specification of crown shapes. Further, methods were implemented to generate different point distributions within the space enclosed by implicit surfaces. Specifically, Lane's distribution method [54, 55] was used to generate distributions based on the user-specified probability density functions, and on the dynamic modification of these density functions. The probability density functions were initialized using the field values of implicit surfaces. This allowed for the generation of point distributions controlled by field values of implicit surfaces. Trees with different branching strucutres were generated by varying point distributions. A particularly interesting case was to limit the point sets outside and in proximity to surfaces, thus causing trees to grow around surfaces, and allowing shapes to be suggested by the surrounding branching structures.

Figure 1.2 summarizes the tree modeling process. An implicit crown envelope is defined within the implicit modeling system, is populated with points, and the resulting tree structures are generated using the SCA.

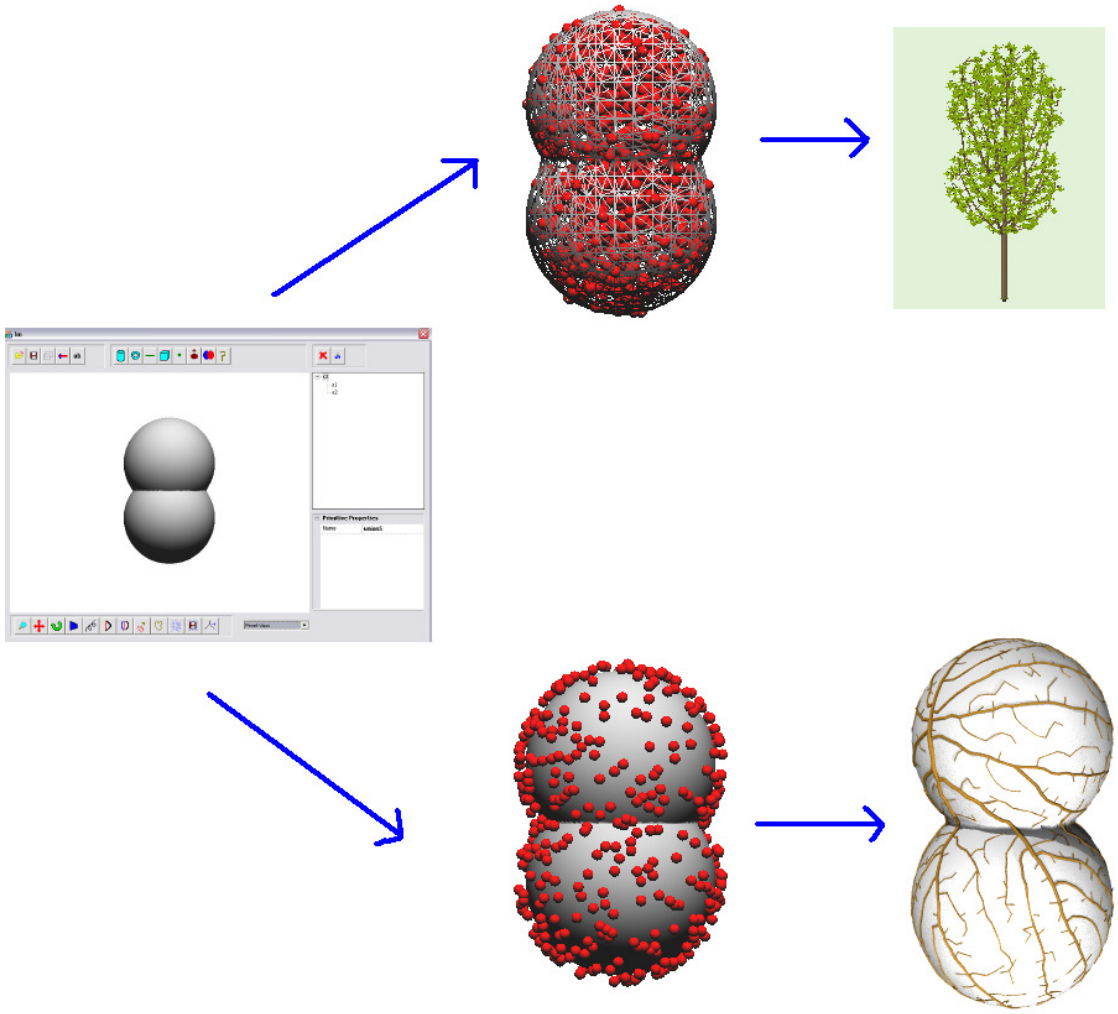


Figure 1.2: The tree modeling process. Left: The implicit modeling system and a crown envelope defined using the modeler. Center: The crown populated with points (both inside the space enclosed by the envelope, and in proximity to it.) Right: The resulting tree structures.

The implicit modeling system was developed to be more powerful than is needed for the specification of crown shapes. Specifically, implicit skinning was implemented. Skinning, also known as lofting, is a popular modeling paradigm in the parametric

domain, but it hasn't caught up in the implicit area. In this technique, the user specifies planar profile curves and a surface is created that passes through these curves. The skinning framework described here uses interpolation of distance fields. Interpolation of distance fields has been used earlier [79, 39]. The contribution here is to adapt this method for implicit modeling purposes. It is shown that by using this paradigm, many complex surfaces can be easily created by the specification of cross-section curves.

1.1 Organization of Thesis

There are eight chapters in this thesis. Chapter 2 provides an overview of tree modeling approaches. Chapter 3 discusses existing approaches to modeling using implicit surfaces. Chapter 4 describes the implementation of the implicit modeling system that was developed for this thesis. Specifically, implicit skinning framework is presented. Chapter 5 examines existing techniques for generating point distributions. Chapter 6 gives details of the algorithms that were used for generating various point distributions. Also, methods to generate points in proximity to arbitrary triangular meshes are described. Chapter 7 examines the modeling of trees using the system developed for this thesis. Finally, in Chapter 8, I provide conclusions and discuss the results.

Chapter 2

Tree Modeling - Overview

The following sections provide an overview of existing approaches to plant modeling.

2.1 Recursive Branching Structures

Honda introduced trees as recursive branching structures characterized by a small number of geometric attributes such as branching angles and length ratio of internodes [43]. Aono and Kunni adapted Honda's model to computer graphics [7].

Mandelbrot introduced the concept of fractals and their use in the geometrical representation of nature [61]. Inspired by this notion, Oppenheimer proposed fractal tree models [69]. His procedure recursively calls itself to generate child branches along main branches. The tree structure was controlled by parameters such as branching angles, minimum branch size, and size ratio of parent and child branch. In order to produce irregular structures, the parameters were varied throughout the tree by using random numbers.

There exists a relationship between fractals and plants treated as recursive branching structures [61]. The models of Honda [43] and Aono and Kunni [7] can be regarded as fractals. Similarly, the trees used by Bloomenthal [12] have a fractal character.

De Reffye et al [23] proposed modeling of trees using statistical description of the fate of buds — a bud may produce branches or flowers, it may become an internode

or die.

2.2 Global-to-local models

The models listed so far can be described as local-to-global models. In these models, the global form of a tree is the result of local growth rules. It is, however, difficult to specify global aspects such as the overall silhouette of a plant and the shape of a plant axes. An alternative approach is global-to-local models in which the global structure is used to instantiate local characteristics.

Reeves et al [80] introduced a model in which the user specified the shape of the crown. Based on this shape and other user-specified parameters, the algorithm selected characteristics of the tree such as branch length, branching angle, and distance between consecutive branches.

In the model proposed by Weber and Penn [97], a set of textually edited parameters specified the crown shape and geometric parameters for each branching level of the tree. They then inferred the branching structure from this information and also restricted the tree to grow within the bounds of the crown.

Lintermann and Deussen created xfrog modeling environment [26, 66]. It allows users to graphically specify functions that map positions along an axis to attribute values such as length of an internode, the branching angle, and the length of a branch.

Prusinkiewicz et al identified mathematical foundations for global-to-local modeling of plants [78]. Global aspects can be specified by the user and the procedural approach uses this information to instantiate local mechanisms such as the placement of organs along the axes.

2.3 Internal Processes

The development of models in this classification is regulated by the information propagation within the tree structure. This flow of information within a tree is termed endogenous information flow.

Borchert and Honda presented a model in which the development of a tree was controlled by the flux of a substance such as water to different parts of the tree [16]. Branches competed for this limited resource. The flux from the mother branch to the daughter branches was distributed asymmetrically. If the flux allocated to a branch was greater than a minimum specified threshold, then this branch continued to grow. Otherwise, it died.

A common method for specifying growth and development of plants is L-systems, introduced by Lindenmayer [56]. L-systems is a parallel string rewriting system that represents the growth of an object such as a tree. The rewriting process is controlled by a set of production rules.

Prusinkiewicz, Lindenmayer and Hanan [77], based on biological studies [33, 34, 32, 31, 46], used signals to model growth and development in herbaceous plants [77]. They presented two models, both specified using L-systems. In the first model, a signal is sent from the base to the apex. The signal stops all axis development and causes production of flower upon reaching the apex. In the second model, several signals are used. Before the first signal is sent, further development of the lateral axes is suppressed. The first signal S_1, a flowering signal, is sent from the base to the top along the main axis. When this signal reaches the apex, it results in the initiation of development of terminal flower. Simultaneously, a second basipetal signal S_2 that

enables the growth of lateral axes is sent down the main axis. After some delay, a secondary basipetal signal S_3 is sent along the main axis. Its effect is to send first flowering signal S_1 along lateral axes as they are encountered on its way down to the base. If S_2 travels faster than S_3, that causes lower axes to grow longer than the upper ones. Chiba et al presented a related model [18, 19].

Allen, Prusinkiewicz and Dejong [5] introduced a model of peach trees which used interactions between sources and sinks of carbohydrates. L-systems were used for specifying and implementing this model. The accumulated carbohydrates in the leaf (due to photosynthesis) can flow into various sinks within the tree (root, fruits or stems). If the supply of carbon is insufficient at the organs (fruits, leaves or branches), then the organ is shed. Stem segments, in addition to being sources and sinks, conduct fluxes throughout the tree. The magnitude of the flux depends upon the resistance of the intervening path and upon the difference in the carbohydrate concentrations between sources and sinks. In order to compute the accumulation, flow and partitioning of carbohydrates between the components of the tree, an analogy with electric circuits is used and hence correspondence between biological and electrical quantities is made. For instance, the amount of mobilized carbohydrates correspond with electric charge. The equations that thus arise are solved numerically in L-systems.

2.4 Environmental Influences

In a bid to create more diverse and realistic trees, researchers incorporated the effects of environmental influences on the growth and architecture of trees.

Arvo and Kirk [8] used ray tracing for sensing surroundings. In their work, particles travel through the space by casting rays into the environment and the returned intersection information is used to guide subsequent development. The path of a particle corresponds to a branch or a leaf. By seeking locations that satisfy predetermined environmental criteria such as proximity to surfaces and availability of light, they were able to simulate a number of phenomena such as ivy and grass growing among obstacles.

In a different procedure, Chiba et al considered the effect of light on branch shedding and on the orientation of branches towards the brightest directions [19, 18]. They considered the amount of light falling on a cluster of leaves belonging to a given branch which they termed as a leaf-ball. To do this computation, they used the projection of a leaf-ball (i.e, a cluster of leaves) on a sphere, and estimated the area of the shadows projected from the other leaf-balls. If the percentage of the light that reached the leaf-ball was less than a given threshold, then they removed the corresponding branch as a withered branch. Takenaka proposed a related model in which the growth of branches and the fate of buds depend upon the local light environment [91].

Prusinkiewicz, James and Mech introduced an extension to L-systems for simulating the impact of environment on plant development [74]. They applied the system to simulate plant's response to pruning and created models of plants sculptured into various interesting shapes represented, for example, by implicit surfaces.

The above model was extended by Mech and Prusinkiewicz [65]. They introduced a modeling framework in which a plant and the environment are treated as two separate processes interacting using a bi-directional communication channel. For

this purpose, they introduced the language of open L-systems that could be used to exchange information between the plants and their environment. They presented models that demonstrated the development of roots in the soil, the propagation of clonal plants, collision between the branches and the development of tree crowns competing for light.

2.5 Competition for Space

In this class of models, branches compete for space and this competition between the branches for space determines the tree structure. The idea of competition for space can be traced back to Ulam, who, in 1962, created a 2D cellular-automation of tree-like branching structures [94].

Ned Greene proposed a model in which 3D space is divided into voxels possibly containing geometrical structures [36]. The voxels that are intersected by objects and plants are marked as occupied. Plants grow towards voxels that are unoccupied. Plants can be grown in proximity to surfaces by ensuring that a new node is not created in a voxel whose distance to the nearest object is greater than a given threshold. He incorporated the effect of light in his model by voxelizing the three dimensional space. He computed light exposure at each voxel by casting rays towards the sun and the sky. The fraction of occluded rays determined the sun and sky exposure at each voxel. To grow plants towards brighter areas, a node was prevented from growing in a voxel that had sun and sky exposure less than a minimum specified value.

Benes and Millan used the approach suggested by Ned Greene in their work on climbing plants [10]. They used particles to represent the active and dormant buds.

The active particles seek their path through the environment which is voxelized. Particles use available light and proximity to objects to select the best available path. Benes and Millan incorporated the concept of inhibitors in their model. If the active bud cannot proceed further, which might happen when it encounters obstacles in its path, the bud activates a dormant bud down the path before aborting.

Rodkaew et al's model works by distributing particles within a crown [81]. A target point is specified at the bottom of the crown. In an iterative procedure, particles move towards the average direction of its nearest neighbor and the target. The path of the particle becomes a branch. When two particles are close enough they are joined to form a parent branch. Each particle has an energy representing branch width. When two particles join, their energies are also combined so that the parent branch is thicker than the child branches. The simulation proceeds until all the points have reached the target point.

Neubert et al [66] presented a model which builds on the Rodkaew's model [81]. The input in their model is a set of photographs of a tree taken from different views. They then remove background from the photographs and build a voxel-model of the tree volume. Each voxel contains the estimated density of the tree biomass. They then generate the particles within the voxels based on the density in each voxel. The simulation proceeds by directing the particles to move towards the root as in Rodkaew et al's model [81]. The directions of the particles, however, are modified so that the generated tree structure is similar to the input photographs. To achieve this, they construct two-dimensional attractor graph for each input image by inspecting the trunk and main branches present in the photograph. These attractor graphs are combined to modify the directions of the particles. In both the approaches by

Rodkaew et al and Neubert et al, the tree grows from the branch tips to the tree base.

In the model presented by Runions, Lane and Prusinkiewicz [83], a given crown is populated with marker points. These points mark the availability of free space. From a given start node at the tree base, the tree grows in an iterative fashion. In each iteration, each attractor point selects the closest tree node. Hence, for each tree node v, there is a set $S(v)$ of associated attractor points. If $S(v)$ is not-empty, a new node v' is created and attached to v. The node v' is positioned in the direction defined by averaging all the normalized directions from v to each attractor point in $S(v)$. Then those attractor points are removed from the simulation that are with in a specified distance from any tree node. Figure 2.1 summarizes the operation of the algorithm.

2.6 Modeling Ramiforms

The previous sections reviewed methods for the modeling of tree structures. Once we have a tree structure, we need to compute branch widths, and model tree limbs and their connections ("ramiforms") respecting these widths. A popular method for computing branch widths is using the pipe model [89, 70]. The model assumes that each leaf is connected to the base of the tree by a pipe of constant radius, and that the number of pipes running through a branch contributes to the width of the branch. If two branches with radius $r1$ and $r2$ meet at a branching point, then the radius r of the supporting branch is computed as

$$r^n = r_1^n + r_2^n \tag{2.1}$$

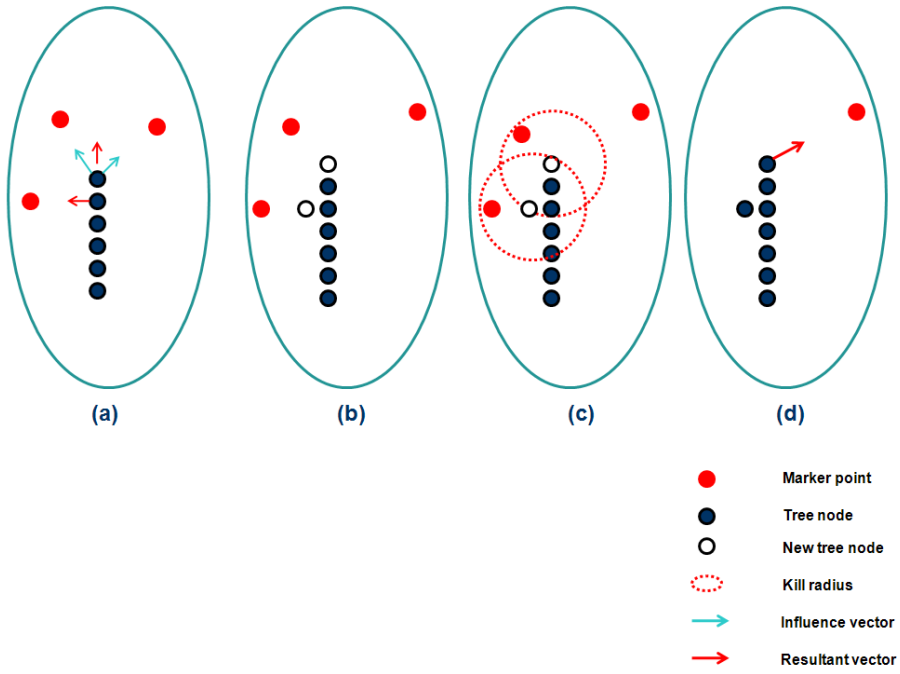


Figure 2.1: Space-colonization algorithm. a) Marker points (red disks) influence closest tree nodes (blue disks); b) New tree nodes (black disks with white centers) are created in the resultant influence direction (red arrows); c) Marker points within the kill radius of new tree nodes are removed from the simulation; d) The simulation continues with the remaining marker points, and new tree nodes.

where n is a parameter (usually between 2 and 3, see [59]). The branch width computation proceeds from the branch tips which are assigned some radius r_0 to the base of the tree.

Holton [42] introduced a plant modeling method that was inspired by the pipe model. In this model, strands run from the base of the tree through its branching structure. A probabilistic model distributes the strands of the parent branch between the child branches. Branch characteristics such as the branch thickness, branching angle and the length can be specified as the functions of number of strands at that branch. The width of each branch is proportional to the square root of the number

of strands in that branch. The relationship between the diameters of the parent and child branches is given using equation 2.1 with $n = 2$.

Once branch widths have been computed, we need to model tree limbs and their connections ("ramiforms") that respect these widths. Bloomenthal [12] used generalized cylinders to represent branches. In his model, branches were specified as a list of points. Cubic interpolating spline was used to connect these points. A generalized cylinder was constructed by sweeping a closed 2D curve along the cubic spline. Bloomenthal presented a method to create a smooth surface ("ramiform") at the branching point. The ramiform was constructed by joining the circular cross-sections at the end of each branch by cubic spline curves and joining these curves to form a polygonized representation.

The method presented by Bloomenthal is not easily extended to arbitrarily complex ramiforms. Furthermore, it creates a smooth junction between branches. Many tree types exhibit non-smooth features such as branch bark ridges and collars. Galbraith, MacMurchy, and Wyvill addressed these issues in their work on BlobTree Trees [35]. They used skeletal implicit surfaces to represent branches. To simulate non-smooth features at the branching points, they deformed the child branches using precise contact modeling (PCM) (discussed in Section 3.3), and then blended with the parent branch. To reduce bulging — a problem commonly observed when blending implicit surfaces — they displaced away the branches from the branching point before blending them.

MacMurchy described a method to form a smooth surface at the branching point using subdivision surfaces [60]. In this method, the length of the branches at the branching point are adjusted so that they do not intersect with each other. An initial

coarse mesh is generated for the unbranched portion of the mesh. A coarse mesh can be generated either as a closed volume (for trunks and stems) or as an open surface (for compound leaves). At the branching point, different branches are connected by inserting junction structures. The completed coarse mesh is subdivided using Loop's scheme. There are three types of ramifications supported by this method: a single branch positioned laterally with respect to the main branch, symmetric bifurcation, and branches on both sides of the main axis (trifurcation).

2.7 User control of the models

In any modeling environment, the ability to interact with the system plays an important role. This interaction may be as simple as textually specifying parameters, or more advanced, using a graphical interface. Oppenheimer used sliders in his system [69]. Weber and Penn specified parameters in text files [97]. Mercer, Prusinkiewicz, and Hanan [63, 76] created a virtual laboratory for L-systems that lets users customize control panels for graphically manipulating diverse models.

Deussen and Lintermann created the xfrog modeling system [26, 66]. It is an interactive plant modeling system which allows users to graphically edit function plots, which specify functions that map positions along an axis to attribute values such as the length of an internode, the branching angle, and the length of a branch.

Okabe, Owada, and Igarashi presented an interactive system in which the user sketches a tree in 2D and the system converts it to 3D [68]. They perform this operation using the assumption that the trees spread their branches so that the distance between each branch is as large as possible. Further, once the tree has been

constructed, the user can sketch a new trunk and a silhouette. The system changes the shape of the trunk and adjusts the length of each segment of the tree to fit it into the silhouette.

Ijiri et al. also presented a sketched-based system for flower modeling that transformed an initial sketch to a 3D representation [44]. The user sketches 2D strokes representing the overall appearance of the model. The strokes are created on hierarchical *billboards* — each billboard representing the type of a component the user wishes to create such as branch, leaf, or flower. The user can then refine each sketch by adding more detail. Finally, each sketch is replaced by a 3D component.

Ijiri et al presented a sketch based system that encapsulated L-systems behind its interface [45]. It allowed a user to sketch the central axis of a plant and generated the whole structure around it. By varying the central axis curve, the user could generate different global shapes that had the same branching topology. The system also presented the L-system rules graphically, and allowed the user to modify the parameters such as the length and the width ratio of the branches, twisting angle etc.

Anastacio et al presented a sketched based system for modeling of single-compound plant structures with phyllotactic arrangement [6]. In their model, a 3D representation of a plant is constructed from a set of 2D strokes. For instance, the user draws a curve to represent the main plant body (stem), and lateral strokes across the main plant body to represent the inclination of organs. Similarly, the user can specify shape of organs by sketching in 2D. From this set of strokes in 2D, a 3D representation is constructed and organs are placed in phyllotactic pattern.

Wither et al introduced a system in which the user sketches 2D silhouettes of the

tree foliage at multiple scales, and the structure of the 3D tree is inferred from this knowledge [99].

Palubicki et al [71] extended the space-colonization algorithm [83] by interactively manipulating the environment in which the tree grows. They used a procedural 3D brush that dynamically creates markers of free space. The brush is controlled by a pressure-sensitive tablet pen. Light pressure is used to sketch individual limbs of the tree, while heavier pressure yields entire branches or even the whole tree.

2.8 Summary

This chapter provided a review of the existing approaches for modeling trees. In this thesis, the space colonization algorithm is used to grow plants. The crown shape for these plants is defined using an implicit modeler. A review of existing modeling approaches using implicit surfaces is presented in the next chapter.

Chapter 3

Implicit Surfaces - Overview

This chapter provides an overview of modeling approaches using implicit surfaces. In this thesis, implicit surfaces have been used for the interactive specification of crown shapes. Consiquently, this chapter focuses on basic techniques of modeling implicit surfaces that are relevent for the specification of crown shapes.

3.1 Definition

The equation $f(x, y) = c$ *implicitly* defines a curve in 2D. The curve is a set of points that satisfy the above equation.

This concept extends to higher dimensions. The equation $f(x, y, z) = c$ is the implicit equation of a surface. The set of points $p(x, y, z)$ that satisfy the above equation form the implicit surface. For example, the equation below represents a sphere of radius r

$$r^2 - (x^2 + y^2 + z^2) = 0. \tag{3.1}$$

The function f can more formally be represented as $f : R^3 \rightarrow R$. It assigns a scalar value to a given point. If the implicit surface is closed, then the sign of the returned value indicates if the point is inside, outside or on the surface. For example, in equation (3.1), a positive value indicates that the point is inside the surface, while a negative value indicates that the point is outside, and zero value indicates that the point is on the surface. The function f is known as a *field function*. The value $f(p)$

is known as a *field value* of the given point p.

The points that belong to an implicit surface are not known and have to be found by employing some search strategy in R^3. Interactive visualization (Section 3.4) of implicit surfaces is an active area of research in the field of implicit modeling.

3.2 Distance Surfaces

The implicit equation defined in equation (3.1) assigns a value to each point $p(x, y, z)$ of R^3. A particular way of defining implicit surfaces is using composition:

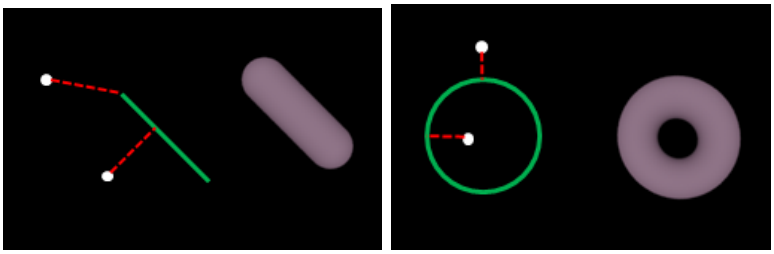
$$(f \circ d)(p) = f(d(p)). \tag{3.2}$$

The function $d(p)$ is a distance function – it is the distance of a given point p to some entity known as a skeletal element. The entity may be a point, a line, a curve or a surface. The implicit surface, known as a distance surface, is defined as the set of points that, for a given scalar value v, satisfy the implicit equation $f(d(p)) = v$. v is known as the *iso-value*. Figure 3.1 shows line and circle skeletons and corresponding implicit surfaces.

This approach of using distance functions for modeling was introduce by Blinn in 1982 [11]. He was motivated by the problem of displaying molecular structures. Existing models of molecular structures consisted of a collection of intersecting spheres and cylinders. Using this approach, he was able to model smooth bonds between atoms. For each atom, he defined the field function as

$$g(p) = b \exp(-ar^2).$$

where r is the distance of a point p to the center of the atom and a and b are user



(a) (b)

Figure 3.1: The skeleton elements and the corresponding implicit surfaces (Distance surfaces). The white dots represent points and the red lines represent the distance of these white points to the skeleton elements. a) A line skeleton element and the corresponding implicit surface; b) A circle skeleton element and the resulting implicit surface (Torus).

defined parameters that control the radius and blobbiness of the resulting surface. The field value of a collection of atoms was obtained by summing the contribution of each atom

$$f(p) = \sum_i b_i \exp(-a_i r_i^2).$$

The field function introduced by Blinn has an infinite support. For purposes of efficiency, it is desirable to have field functions that have compact support — these functions evaluate to zero for all distance values greater than a given threshold R. Consider an implicit surface that is composed from several skeleton elements. To find the field value at a given point p, contribution from only those skeleton elements need to be considered that are within the distance R from a point p. Nishimura [67] and Wyvill et al [103] introduced field functions with compact support.

The field function introduced by Wyvill et al [103] is defined as

$$f(r) = \begin{cases} a\frac{r^6}{R^6} + b\frac{r^4}{R^4} + c\frac{r^2}{R^2} + 1 & \text{if } r <= R \\ 0 & \text{otherwise} \end{cases}$$

where r is the distance of a given point from the skeleton element and R is skeleton's radius of influence (such that $f(r) = 0$ for $r \geq R$). See Figure 3.2. The coefficients $a = -0.44444$, $b = 1.888889$, $c = -2.44444$ are solved by specifying the following constraints

$$f(\frac{R}{2}) = \text{isovalue} = 0.5$$
$$f(R) = 0$$
$$f'(R) = 0$$

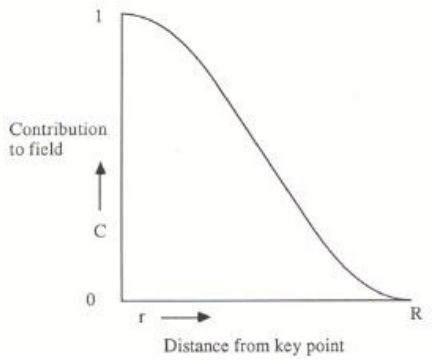


Figure 3.2: Wyvill field function defined in [103]

3.3 Blending

The purpose of blending is to combine implicit surfaces to form a new surface. In this manner, complex implicit surfaces can be constructed from simple ones. The inherent strength of implicit surfaces is that they can be easily blended smoothly. A

particular form of blending is the *summation* blending. Given field functions f_1 and f_2 for two surfaces, the blending between them is performed by adding the two field functions. The field function $f(p)$ of the blended surface is given as:

$$f(p) = f_1(p) + f_2(p)$$

Using this approach, an implicit model can be built hierarchically from other implicit surfaces using blending and other operations such as difference and warping. Wyvill, Galin, and Guy presented such a hierarchical system — the Blob tree — for modeling using implicit surfaces [102]. Figure 3.3 shows the Blob tree for a model constructed using this approach.

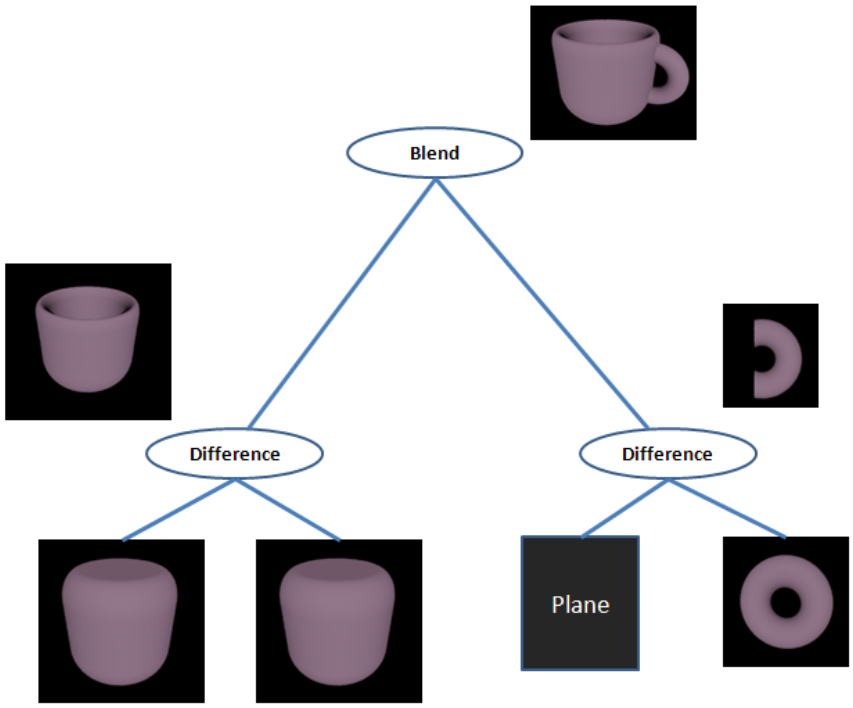


Figure 3.3: The Blob tree [102] for a model constructed from a cylinder and torus using the blending and difference operations.

3.4 Visualization

Visualization of implicit surfaces has received a great deal of attention. The points that belong to the surface are not known and have to be found by employing some search strategy in R^3. Visualization approaches can be classified into two categories — approaches that polygonize the implicit surface and then display this polygonized approximation and the approaches based on ray tracing [38, 15]. In this section, I briefly describe the approaches based on polygonization of implicit primitives.

Wyvill et al [103] and Lorensen and Cline [58] independently described a polygonization approach in which the space is subdivided using cubes. The implicit function is used to evaluate values at the corner of each cube. If some corner of the cube is inside the implicit surface and the other corner is outside the surface, then the surface intersects the cube. The surface passing through the cube is approximated by triangles using a predefined set of rules depending upon the state of the corners. In this manner, the whole surface is polygonized. Wyvill et al [103] optimized traversing through the space by making use of the surface continuity.

Over the years, researchers have proposed modifications and improvements to this approach [14, 15, 49, 48, 41]. With the original approach, creases and corners in the implicit surface can not be found. Wyvill and Overveld presented a polygonization technique that accounted for the CSG operations on implicit surfaces, and consequently preserved creases and corners. Kobbbelt et al introduced Extended Marching Cubes [49] that allows them to find the sharp features in the surface. They compare the normals at the eight corners of the cube. If the normals differ greatly, they conclude that edge or a corner must be present.

3.5 Summary

This chapter provided an overview of modeling approaches using implicit surfaces. The next chapter discusses the features of the implicit modeling system that was developed as part of this work.

Chapter 4

The Implicit Modeling System

This chapter provides a description of the implicit modeling system that was developed for this thesis. The purpose of the modeler is to interactively specify the crown shapes that can be used for growing trees. The modeler is described in the following sections.

This chapter also describes implicit skinning framework. The implicit skinning is performed using interpolation of distance fields. Interpolation of distance fields has previously been done by various authors [79, 39]. In [79, 39], shapes of organs were reconstructed from parallel slices obtained from medical imaging scanners. The contribution here is to adapt distance field interpolation for implicit modeling purposes.

4.1 The Implicit Modeler

The modeling system was developed using C++, OpenGL, and MFC. It incorporates features such as distance surfaces, variational curves and surfaces, implicit surfaces of revolution, CSG, blending, sweep implicit surfaces, and implicit skinning. Further, deformation operations such as bending, tapering, and twisting are defined.

Users have two options to interact with the modeler. Firstly, the user interaction is by means of toolbars, dialog boxes, or a mouse. For instance, the user can drag the mouse to rotate, or translate a surface. Similarly, the user can sketch the cross-sections for sweep and skinned surfaces. Figure 4.1 shows an example. Figure 4.2

shows an example in which a boolean operation between two surfaces is performed by sketching on the surfaces. The type of a boolean operation is selected from the dialog box.

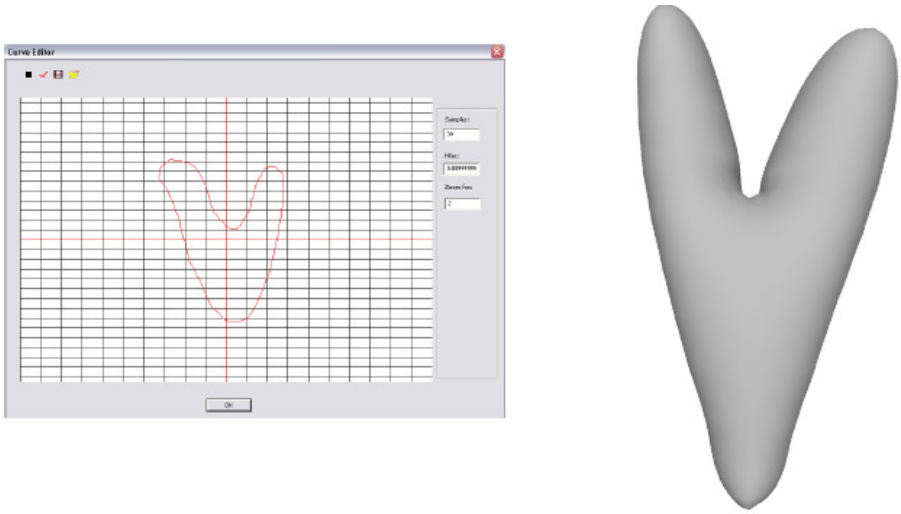


Figure 4.1: Defining a sweep surface by sketching the cross-section curve.

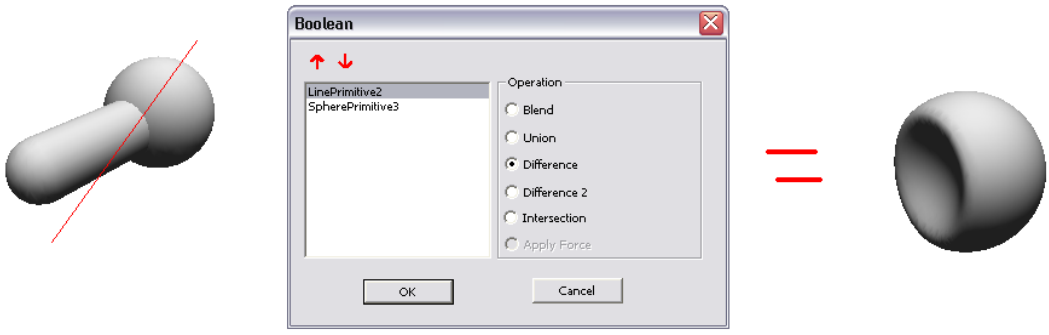


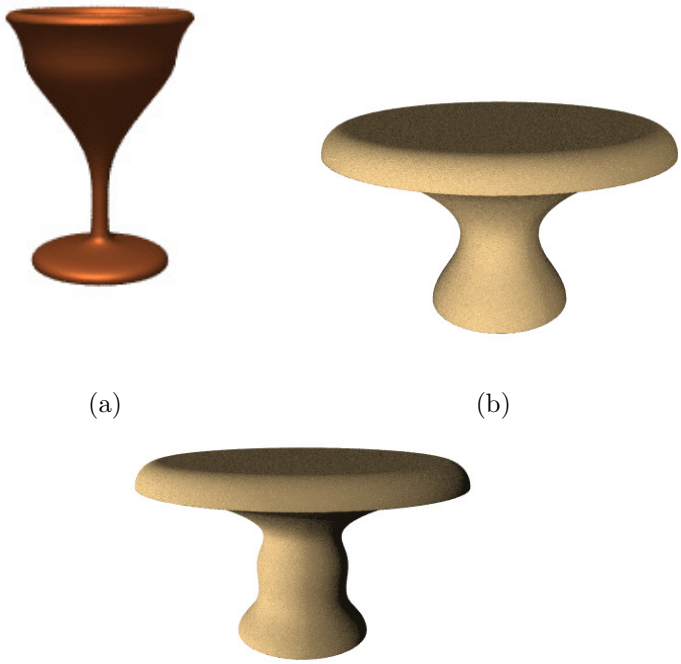
Figure 4.2: Specifying a boolean operation between two surfaces by sketching on them (the red stroke on implicit surfaces), and selecting the desired boolean operation from a list in the dialog box.

The modeling system also provides a command editor, which can be used to give

textual commands to manipulate the models. For instance, to perform the boolean operation of Figure 4.2, the user could give the following commands:

```
sphere s1
line s2
diff s1 s2 s3
```

Figures 4.3, 4.4, 4.5, and 4.6 show some of the models created using the implicit modeling system.



(a) (b)

(c)

Figure 4.3: Implicit surfaces of revolution

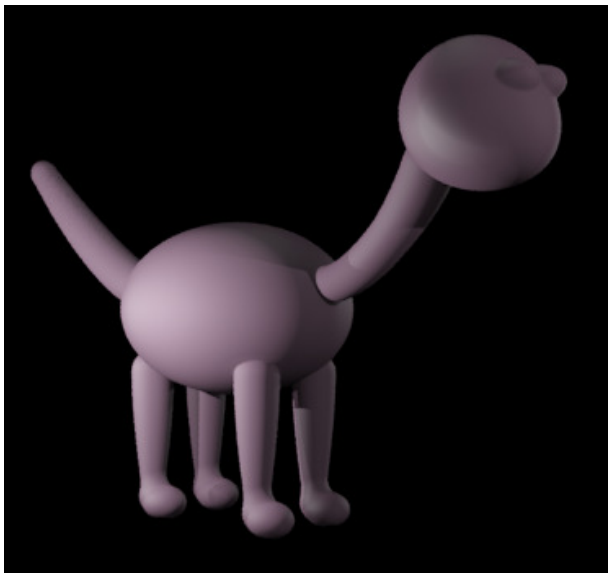


Figure 4.4: A model created using the implicit modeling system.

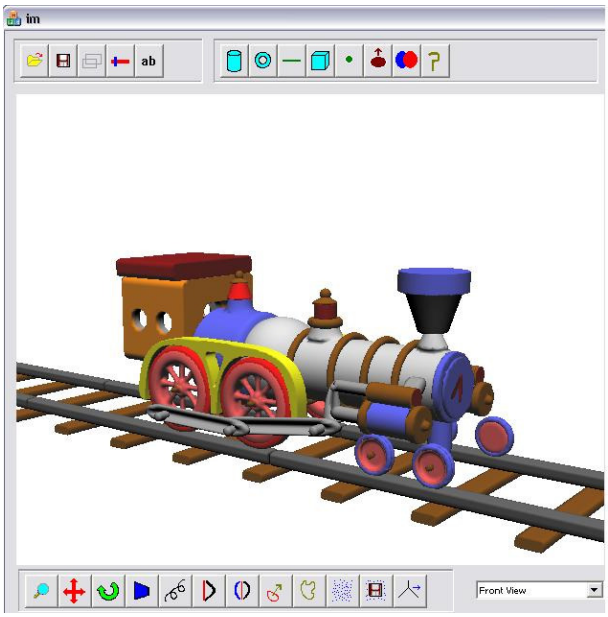


Figure 4.5: A train model created using the implicit modeling system.

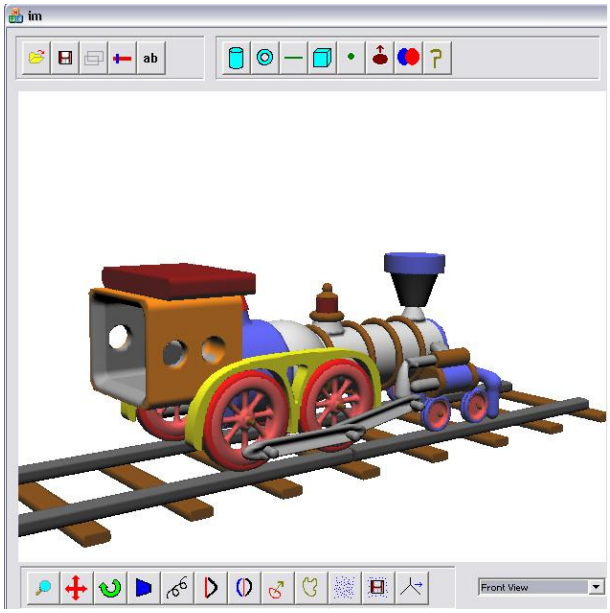


Figure 4.6: The train model from a rotated view.

4.2 Implicit Skinning

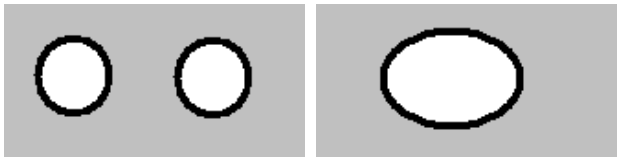
Skinning, also known as lofting, is a very popular modeling paradigm in the parametric domain [100, 101, 72, 73]. In this process, the user specifies planar profile curves and a surface is constructed that passes through these curves. Additionally, as in the case of sweep surfaces [72, 22, 37, 86], a path curve is specified and the surface follows this curve. In the implicit domain, there has been previous work on constructing implicit functions from cross-sections [84, 17, 4, 93], but these methods yield functions that are expensive to evaluate.

The approach presented in this section is to construct skinned implicit surfaces as distance surfaces. As specified in Section 3.2, a distance surface is given by the equation:

$$(f \circ d)(p) = \text{iso-val}$$

where f is a field function and d is a distance function. The field function used was the one presented in Section 3.2 (Figure 3.2). In this section, the construction of the distance function for skinning is presented.

One approach to define the distance function is the following. The cross-sections can be represented as B-Splines, and these B-Splines can be interpolated to form a skeletal element. This, however, results in a distance function that is expensive to evaluate. Secondly, the scenario shown in Figure 4.7 is difficult to handle with this approach. Given curves C1 and C2 in Figure 4.7, how should we interpolate between these curves? The skinning framework presented in the next section handles this case.



(a) (b)

Figure 4.7: For curves $C1$ in Figure a and $C2$ in Figure b, how should we create inbetween curves?

4.2.1 The Method

Given a planar cross-section curve C, and a point p, it is assumed that the distance function for the curve is known, and that the normalized arc-length parameterization of the path curve is given by the function $s(p)$. Secondly, for a given point p on a path curve, the tangent, normal, and bi-normal vectors are denoted as $T(p), N(p)$,

and $B(p)$. These vectors, for instance, can be found by using the rotation minimizing frame of Bloomenthal [13].

4.2.2 Associating cross-sections with the path curve

To create skinned surfaces, the user first specifies planar cross-section curves, and *associates* each curve with a point of the path curve. Figure 4.8 shows the process for two cross-section curves, and the path curve that passes through the points PQR. The rectangular cross-section is associated with the points P and R of the path curve (whose normalized arc-lengths $s(A)$ and $s(B)$ are 0 and 1 respectively). The second cross-section is associated with the point R of the path curve ($s = 0.5$).

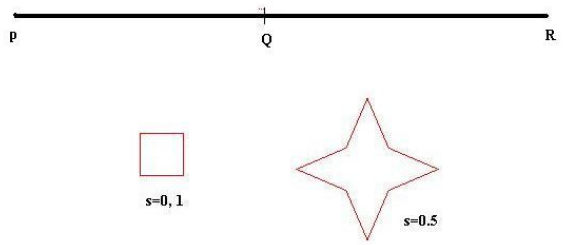


Figure 4.8: Associating planar cross-section curves with the points of the path curve.

Associating a cross-section curve C with a point p of the path curve results in the following. If a plane intersects the skinned surface such that it passes through the point p of the path curve, and T(p) is parallel to the normal of the plane, then the plane intersects the skinned surface along the cross-section curve C.

4.2.3 Distance Field Computation

For a cross-section curve C_i and its associated point q_i on the path curve, denote s_i by $s(q_i)$. Given planar cross-section curves C_1, C_2, ..., C_n with distance functions d_1, d_2, ..., d_n, and a point $p \in R^3$, the distance function $d(p)$ of the skinned surface is defined in the following manner. For a point p, its projection p' on the path curve is determined. The projected point satisfies the following condition, and can be found by using Newton-Raphson method [72]:

$$T(p').(p' - p) = 0. \tag{4.1}$$

The point p lies in the plane formed by the point p' and the vectors $T(p')$ and $N(p')$. Considering that the cross-section curves C_1, C_2, ..., C_n lie in this plane about the point p', the distance function $d(p)$ of the skinned surface is given as:

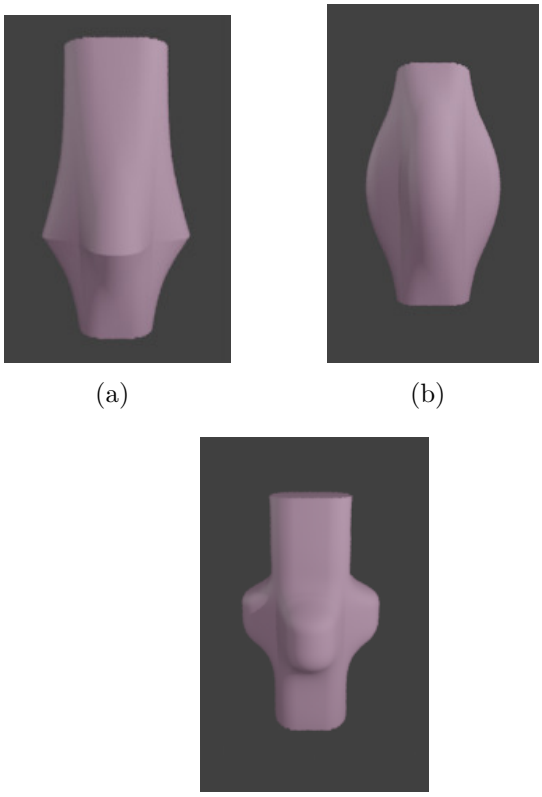
$$d(p) = g(d_1(p), d_2(p), ..., d_n(p), s). \tag{4.2}$$

The choice of the interpolation function g affects the shape of the skinned surface. For instance, for $s(p') \in [s_i, s_{i+1})$, the linear interpolation function is defined as:

$$d(p) = (1 - u)d_i + ud_j, u = \frac{s - s_i}{s_{i+1} - s_i}$$

Figure 4.9 shows various skinned surfaces for the cross-section curves of Figure 4.8.

Figure 4.10 shows an example of an implicit skinned table. As shown in the figure, four cross-sections are associated with the beginning of the path curve and one cross-section for the end. Skinning such models with distance field interpolation is very easy, and would be difficult using traditional parametric skinning methods.



(a) (b)

(c)

Figure 4.9: The effect of different interpolation functions on the resulting implicit surfaces. The cross-sections of Figure 4.8 are used for defining these surfaces. a) Linear interpolation. b) Quadratic interpolation. c) Cubic Interpolation; the rectangular cross-section is repeated thrice near $s = 0$, $s = 1$, and the star cross-section is repeated thrice near $s = 0.5$.

(a) (b)

Figure 4.10: An implicit skinned table defined from five cross-section curves.

Figure 4.11 shows implicit surfaces created from circular cross-sections, and by using different interpolation functions. The surfaces in Figures c and d have been further refined by scaling and subtracting from the original surfaces. As shown in the figures, a wide variety of shapes can be easily achieved by using this technique.

Figure 4.12 shows two cross-section curves and the corresponding skinned surface. The surface is shown from different views. The skinned surface can be further refined by blending operations. Figure 4.13 shows a cross-section of the field values of the implicit surface of Figure 4.12. As shown in the figures, this method for implicit skinning produces smooth and bounded field values.

4.3 Summary

This chapter provided a description of the implicit modeling system that was developed as part of this thesis. Further, the implicit skinning framework was presented. The modeler is used to interactively specify the crown shapes. Once a user defines a crown, it needs to be populated with point distributions. The next chapter provides a review of the methods that generate point distributions.

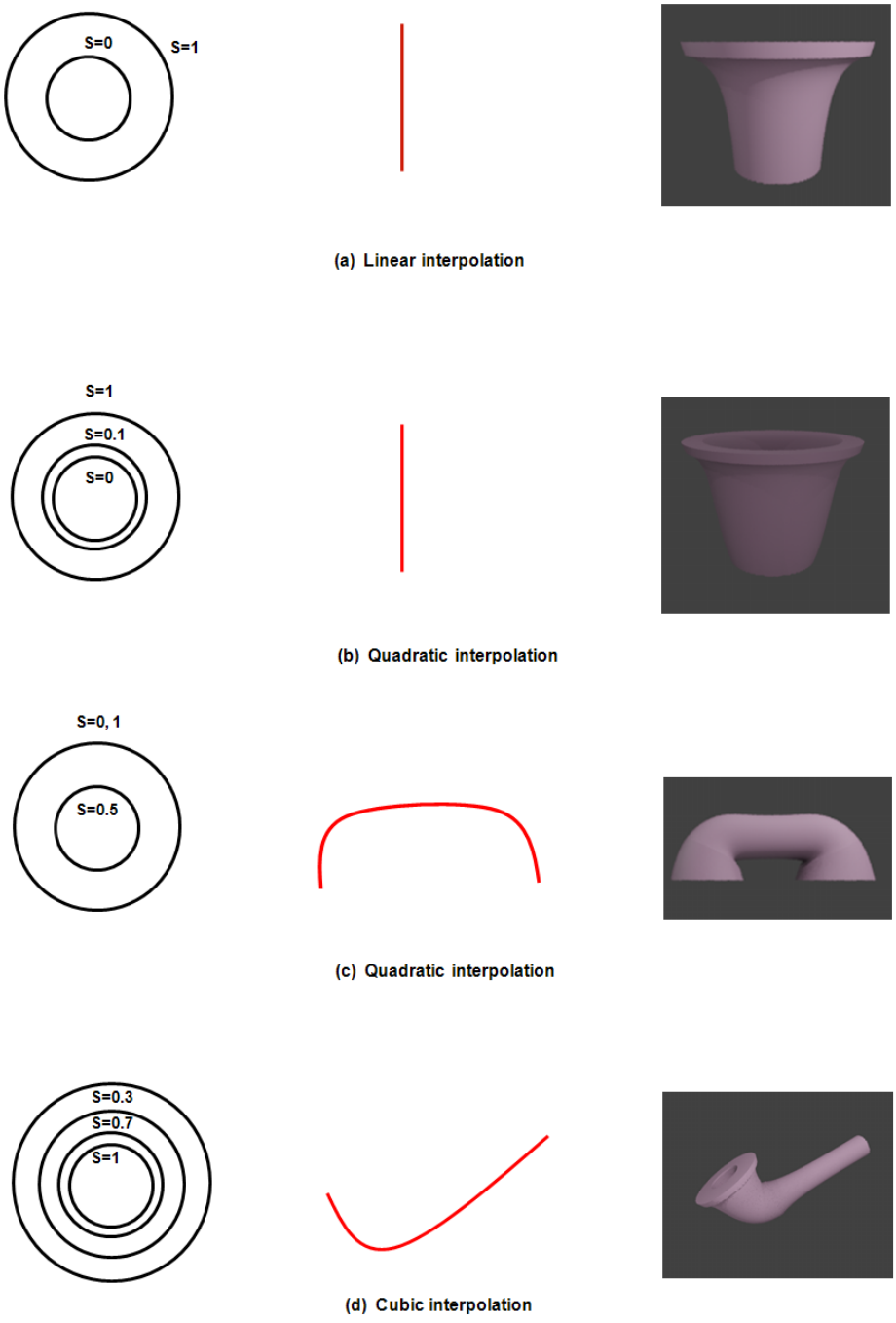


Figure 4.11: Skinning from circular cross-sections. Left: Cross-section curves, and their association with path curves. Center: The path curves. Right: The resulting surfaces.

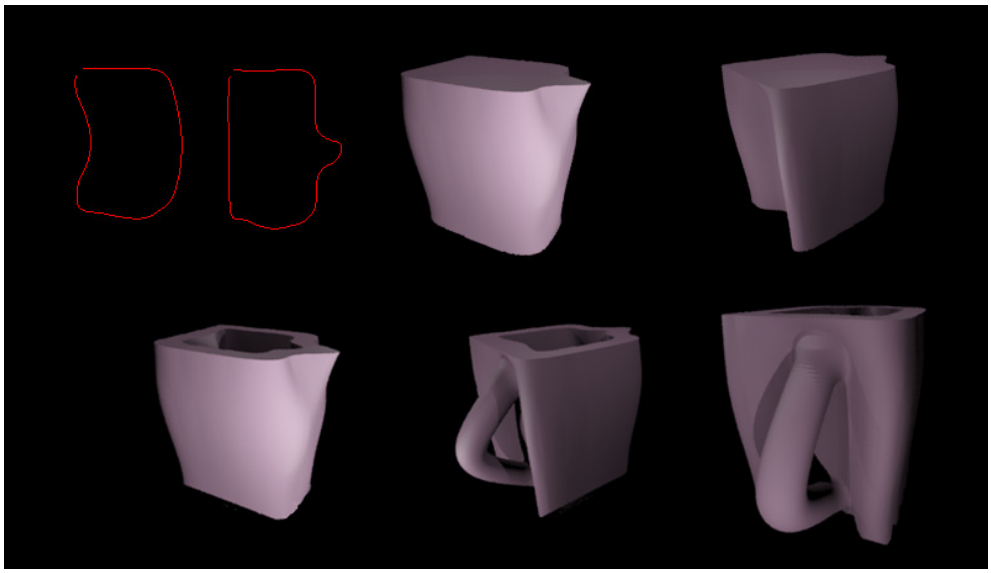
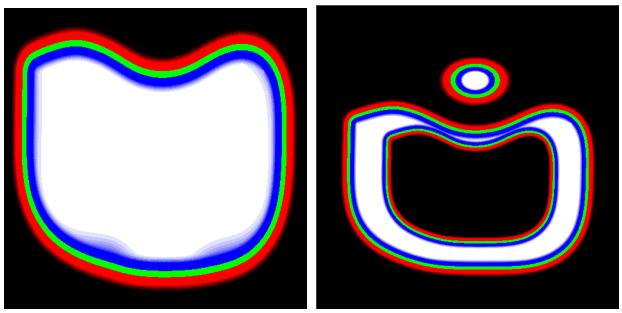


Figure 4.12: An implicit surface skinned from two cross-section curves, and then further refined using difference and blending operations.



(a) (b)

Figure 4.13: Cross-section of field values of the implicit surface of Figure 4.12.

Chapter 5

Point Set Generation - Overview

This chapter provides an overview of point set generation techniques. These point sets are used by the space colonization algorithm [83] for generating trees and for growing structures in proximity to surfaces.

5.1 Sampling Patterns

A sampling pattern is a distribution of points, which is used for sampling a continuous source, such as an area light source, or an image-plane in ray tracing. A sampling pattern may be completely random, as shown in Figure 5.1a. The drawback of this approach is that it may leave large sections of the sampling domain poorly sampled. In contrast to this approach, some sampling patterns work by dividing the sampling domain into smaller regions, and generating a sample for each region (Figure 5.1). The rationale is that this process covers the sampling domain well. In uniform sampling, the sampling domain is divided into a grid of cells and a sample is generated at the center of each cell (Figure 5.1b). Adaptive sampling works by sampling the corners of a cell. If the results returned at the corners differ significantly, then the cell is further subdivided for sampling [98]. In stratified sampling, a sample is generated for each cell of the grid, but is randomly jittered within the cell. Stratified sampling is known to reduce the aliasing errors in ray traced images [21].

Stratified sampling is an instance of stochastic sampling, which was introduced to

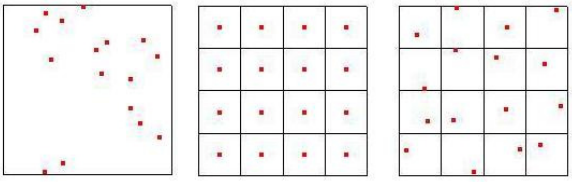


Figure 5.1: Random, Uniform, and Stratified. With random sampling, there may be large areas that are poorly sampled.

computer graphics by Dippé and Wold [27]. They suggested that sampling patterns with blue-noise characteristics in the frequency space would produce better quality pictures and that stochastic sampling could be used to generate such patterns. The sampling pattern has blue-noise characteristic if its spectrum is energy deficient at low frequencies and has no concentrated spikes of energy. They suggested using such patterns since, as they point out, the human visual system is more sensitive to low and mid frequency noise than high frequency noise.

Cook advocated the use of Poisson-disk distributions [21]. He based his suggestion on the study by Yellott [104], which pointed out that the photo receptor cells in the eyes of rhesus monkey are arranged in Poisson-disk pattern and the eyes are very good at avoiding aliasing errors.

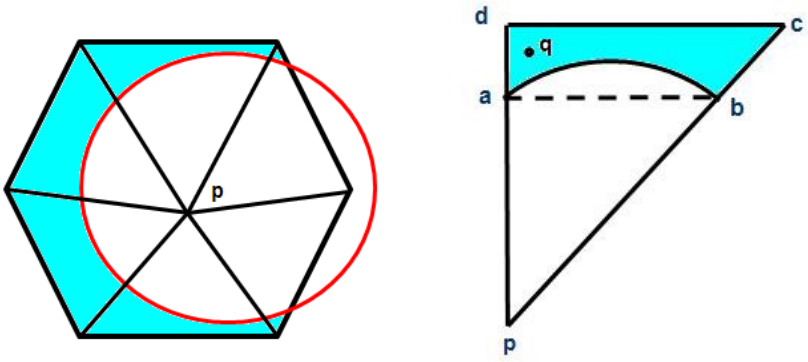
In Poisson-disk pattern, a minimum specified distance is maintained between each pair of samples. Cook generated Poisson-disk samples by dart-throwing algorithm. In this algorithm, distance of the new sample to the already existing point set is computed and the sample is accepted if this distance is greater than the threshold radius [21]. This method is slow, takes long running time and may not sample all available regions of the plane. The process is often terminated after a certain predetermined number of attempts fail to generate a new point. Hence, this process

may terminate before desired number of samples have been generated.

Mitchell introduced a sampling pattern that has properties similar to that of Poisson-disk patterns [64]. To place the $(n+1)^{th}$ point in a point set with n points, mn candidate points are generated, where m is a constant parameter. The candidate point that has the largest distance to the existing point set is selected. Mitchell used grid methods for the nearest-neighbor calculations to speed up the process.

McCool and Fiume generated Poisson-disk pattern by selecting large initial radius [62]. Once no more points could be placed after a certain number of trials, they successively reduced the threshold radius by some fraction. The advantage of this method over dart-throwing is that a Poisson-disk point set with any desired size can always be created. Secondly, once a high density sample distribution has been generated, a lower density distribution can be obtained from it by selecting any prefix of the original sequence. They also adapted the Lloyd's relaxation [57] to improve the spectral properties of the point set. This method repeatedly moves each point of the set to the centroid of its Voronoi region.

Jones presented $O(Nlog(N))$ method to generate Poisson-disk distributions [47]. In this method, a Voronoi diagram of the existing points is created. Then, for each point p of the point set, free space is computed by subtracting a circle of radius r around p from the Voronoi region of p. Figure 5.2 shows the Voronoi region of a point p and the free space (shaded portion). A triangle fan is constructed in each Voronoi region. To generate a new point, a triangle is randomly selected and a point is randomly generated in the shaded portion of this triangle (see Figures 5.2.b and 5.2.c).



(a) (b)

Figure 5.2: a) Voronoi region of a point p. Triangle fan is constructed in the region to compute the area of the free space (shaded portion) and to generate a new point in this space; b) A new point q is generated in the free space of the triangle (the shaded portion) by repeatedly sampling the quadrilateral *abcd* until the distance between q and p is greater than the minimum specified distance.The figures are based on the figures in [47].

Dunbar and Humphreys developed a similar technique for Poisson-disk sampling that maintains a list of available regions [28]. To generate a new sample, an available region is selected and a new point is randomly generated in this region. The available region is defined in the following manner. Let $2r$ be the minimum allowable distance between each pair of samples, and $D(p, r)$ be a disk of radius r around point p. For a sampling domain X and an existing point set P, the available *subdomain* A_x is defined as

$$A_x = X - \bigcup_{p \in P} D(p, 2r)$$

Figure 5.3 shows the available subdomain. The *available neighborhood* (the red region in the figure) of the point set is defined as the intersection of the available subdomain with the union of annuli of radius $2r$ and $4r$ around each point of the point set. A new point is generated in the available neighborhood. The authors presented a

representation of the available neighborhood that allowed them to generate a new point in it in $O(Nlog(N))$ time.

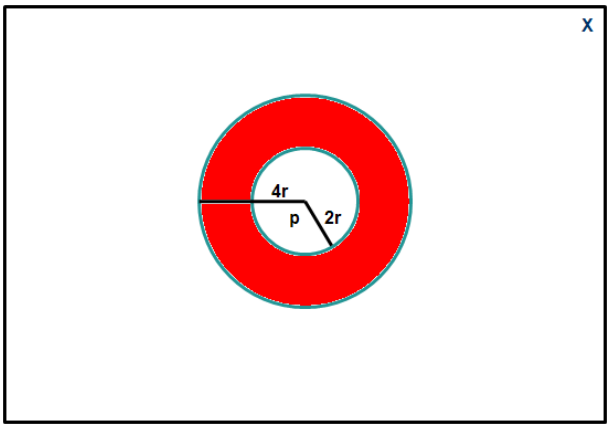
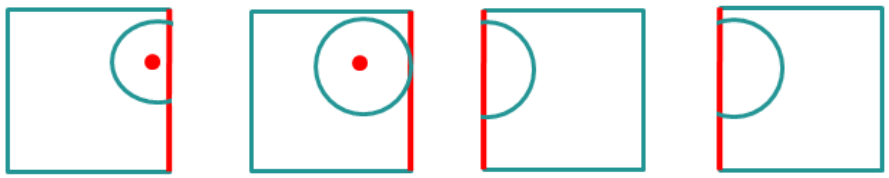


Figure 5.3: Sampling from available region (red region) using the method described in [28]. For a point p, its available region (red region) is the annulus between radii $2r$ and $4r$ around point p. A new point is generated in the available region.

In recent years, several researchers have used Wang tiles [96] for the generation of points with blue-noise characteristics [40, 88, 20, 28, 50]. Wang tiles is a set of square regions with colored edges. Each tile is filled with some data that can be a set of points, texture or geometry [90, 88, 20]. The tiles are joined together in a plane such that the neighboring edges have the matching color.

Shade et al generated Poisson-disk pattern using a set of eight Wang tiles [88]. Poisson-disk pattern in each tile is generated using dart-throwing algorithm. To add a new point, all possible neigbouring tiles are checked. Once all tiles have been populated with points, Poisson-disk pattern for a plane can be generated by stochastically tiling it with Wang tiles. In this method, however, the regions around the edges of the tiles are poorly sampled. Figure 5.4 illustrates the problem.

Hiller et al used Lloyd's relaxation [57] on a set of 13 Wang tiles [40]. Initially,



(a) (b) (c) (d)

Figure 5.4: Poisson-disk pattern using Wang tiles. The region near the edges is poorly sampled. The point near the red edge in Figure *a* has a circle that extends into the tiles in Figures *c* and *d*. If the tile with a point p in Figure b is joined with the tiles in Figure c or d, then the region to the right of the point p will be poorly sampled. The figures have been created on the basis of the figures in [20]

Poisson-disk distribution is generated independently in each tile. Then, a tile is selected and is surrounded by eight other tiles. Voronoi diagram is constructed on these tiles and points in the center tile are repeatedly moved to the centre of their Voronoi region until Poisson-disk constraints are satisfied across the tile boundaries.

The sampling artifacts around the edges of the Wang tiles were corrected by Lagae and Dutré [51, 52]. They used a variation of Wang tiles in which the corners of the tiles are also colored. A tile is divided into three regions: edge region of radius r (where r is the radius of Poisson-disk distribution), corner region and an interior region. To generate samples, first corner regions are sampled. For each corner color, a closed region is formed by combining the corners of four neighboring tiles and samples are generated in this region using dart-throwing. Similarly, the edge regions of the neighboring tiles are separately sampled, thus avoiding the edge artifacts of Figure 5.4. Finally, for each tile, its interior region is sampled by taking into consideration the points in the edge and the corner regions.

Lagae and Dutré have also presented a review of methods to generate Poisson-disk

patterns [53].

Balzer, Schlömer, and Deussen [9] presented a variant of Lloyd's method [57] for generating point distributions. In this method, the points of the distribution are called *sites*. A sampling domain X is discretely represented by m points. The density of these points in any region is proportional to a given density function. Given a random distribution of n sites, the method works by randomly assigning $\frac{m}{n}$ points of X to each site. The assignment is then optimized by swapping points between the sites until each point is assigned to its closest site, thus forming Voronoi regions. Finally, each site is moved to the center of mass of its assigned points. The method results in the improved blue-noise characteristics, prevents the emergence of irregularities such as hexagonal structures of Lloyd's method, and adapts to a given density function.

5.2 Sampling from a Probability Distribution

This section presents methods for drawing samples from a given probability distribution.

Deussen et al. presented a method to control the density of plant distribution by defining a set of gray-level images using a paint program [24]. Given a gray-level image, a point distribution is generated using Floyd-Steinberg half-toning algorithm [30]. Half-toning [95] is a process which is used for the reproduction of a continuous image on a bi-level device by distributing points varying in size or spacing. Deussen et al. [25], Secord [87] and Balzer, Schlömer, and Deussen[9] presented related approaches for the stippling of images.

A particular method for drawing samples from a given probability density function is the inverse cumulative distribution method [29]. This method uses the concepts from the probability theory. Hence, a review of relevent concepts from the probability theory is presented in the next section, followed by the description of the method.

5.2.1 Probability Theory Review

Given continuous random variables [82] X and Y and a joint density function $f(x, y)$, the joint distribution function $F(x, y)$ is given by

$$F(x, y) = Pr\{X <= x, Y <= y\} = \int_{-\infty}^{y} \int_{-\infty}^{x} f(x, y)\, dx\, dy,$$

The joint distribution function satisfies the following property

$$F(\infty, \infty) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y)\, dx\, dy = 1.$$

The conditional density of X given $Y = y$ is given by

$$f(x|y) = \frac{f(x, y)}{f(y)},$$

where $f(y)$ is the *marginal* density function and is obtained by integrating along $Y = y$ line:

$$f(y) = \int_{-\infty}^{\infty} f(x, y)\, dx.$$

The conditional probability distribution of a random variable X given a fixed value of Y is

$$F(x|y) = Pr\{X <= x|Y = y\} = \int_{-\infty}^{x} \frac{f(x, y)}{f(y)}\, dx.$$

5.2.2 Inverse Cumulative Distribution Method

This method is used for drawing a sample from a given probability density function (pdf) $f(x)$ [82]. A normalized function $p(x)$ is obtained from $f(x)$:

$$p(x) = \frac{f(x)}{\int_{-\infty}^{\infty} f(x)\,dx.}$$

Given a cumulative distribution function $F(x) = \int_{-\infty}^{x} p(x)\,dx$, a random variable $Y \in [0, 1]$ is drawn from a uniform distribution. Then, $X = F^{-1}(Y)$ is the required random variable from the given pdf $p(x)$ (See Figure 5.5).

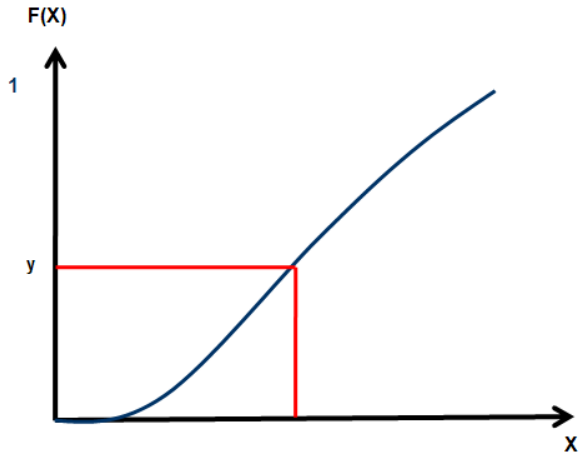


Figure 5.5: Drawing a random variable from a given pdf $f(x)$: Random variable Y is drawn from a uniform distribution. Then $F^{-1}(Y)$ is the random variable drawn from $f(x)$.

A useful technique is to represent the probability density function as a step function [29]. In this case, the computation of cumulative distribution function F and the inverse function F^{-1} is very easy, and consequently it is easy to draw samples from this function. Suppose f is a step function defined over the interval $[a, b]$. The interval is divided into n equal-sized intervals, and the value of f over each of these

equal-sized smaller intervals is constant, and is c_0, c_1, ..., c_{n-1}. The size of each smaller interval is $\Delta x = (b-a)/n$. Suppose, a given x belongs to the i_{th} interval, then the cumulative distribution function F(x) is defined as:

$$F(x) = \frac{\int_a^x f(x)\,dx}{\int_a^b f(x)\,dx} = \frac{\sum_{j=0}^{i-1} c_j \Delta x \; + c_i(x - i\Delta x)}{\sum_{i=0}^{n-1} c_i \Delta x}$$

The inverse function F^{-1} can be easily solved by noting that F is a monotonically increasing function (see Figure 5.6). Suppose a given $y \in [F_{i\Delta x}, F_{(i+1)\Delta x})$, then $F^{-1}(y)$ belongs to the i_{th} interval, and the value of $F^{-1}(y)$ can be found by linear interpolation:

$$F^{-1}(y) = i\Delta x + \frac{y - F_{i\Delta x}}{F_{(i+1)\Delta x} - F_{i\Delta x}}$$

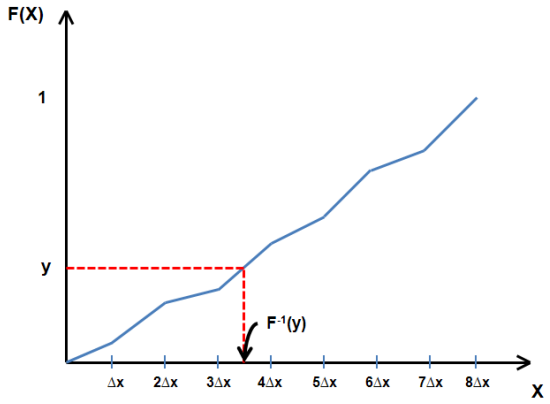


Figure 5.6: The cumulative distribution function F(x) of a step function is a piecewise linear function. Given a $y \in [F_{i\Delta x}, F_{(i+1)\Delta x})$, $F^{-1}(y)$ can be solved using linear interpolation.

The extension of the inverse cumulative distribution method to two-dimensions is as follows [82]: Suppose we wish to find the values of jointly continuous random variables X and Y, given a joint density function $f(x, y)$. We first compute $F(y)$,

which characterizes the probability that the value of the random variable Y is less than or equal to y:

$$F(y) = Pr\{Y <= y\} = \int_{-\infty}^{\infty} \int_{-\infty}^{y} f(x, y)\, dy\, dx.$$

Given $F(y)$, we can find the value of the random variable Y using the function $F^{-1}(u)$ for a uniform random variable $u \in [0, 1]$. To find the value of the random variable X, we use the conditional cumulative distribution $F(X|Y)$.

In the case of $2D$, similar to $1D$, it is easy to draw samples from a probability density function that is represented discretely. In this case, the domain of the density function is divided into a grid of cells of equal size, and the probability density function is constant over each cell. The computation of cumulative distribution function and the inverse cumulative distribution function is similar to the case of one-dimension discussed above.

Based on the inverse cumulative distribution method presented in the previous section, Lane [54] and Lane and Prusinkiewicz [55] presented a method to generate point set distributions in a plane. In this method, probability density function is modified dynamically by the user-specified kernel function. At each step, when a new point is placed in the plane, the probability density field (pdf) around that point is deformed by multiplying it with the kernel function. Hence, the placement of a point in the plane modifies the probability that another point will be found at a given distance from it.

Figure 5.7 shows a kernel function that sets the probability in the immediate neighbourhood of the generated point (red square) to zero. Figure 5.8 illustrates how point distributions are obtained using this approach, and the kernel function

of Figure 5.7. Initially, when no point is generated, we have a constant probability density field (Figure 5.8a). When a point is generated, the kernel function is used to deform the probability field around the generated point (Figure 5.8b, c, and d). The dark blue disks represent the regions with zero probability. Using this process, eventually Poisson-disk pattern is obtained.

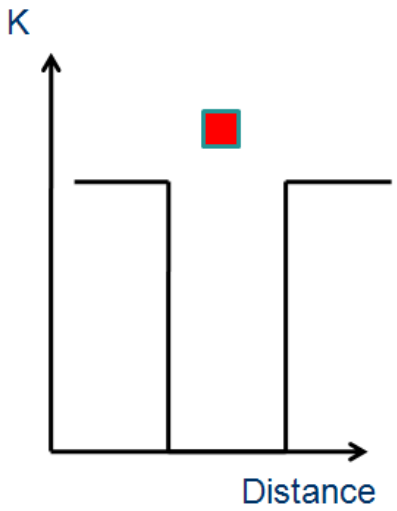
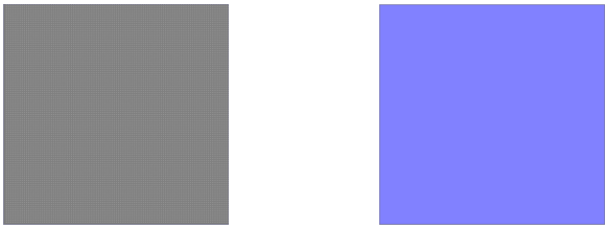
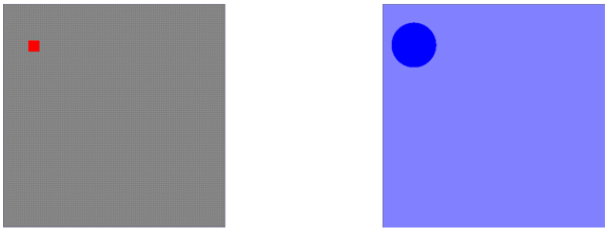


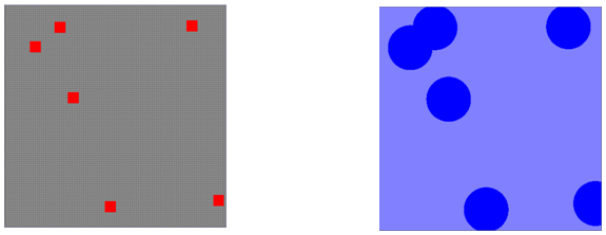
Figure 5.7: A kernel function that sets the probability in the immediate neighbourhood of the generated point (red square) to zero.



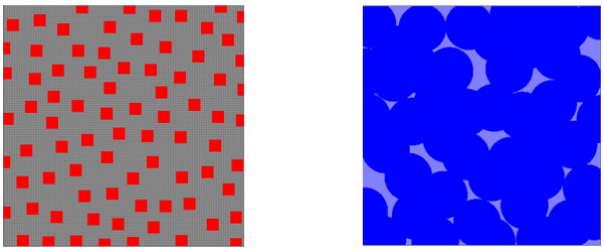
(a) A constant probability density function.



(b) After generating one point and deforming



(c) After generating several points

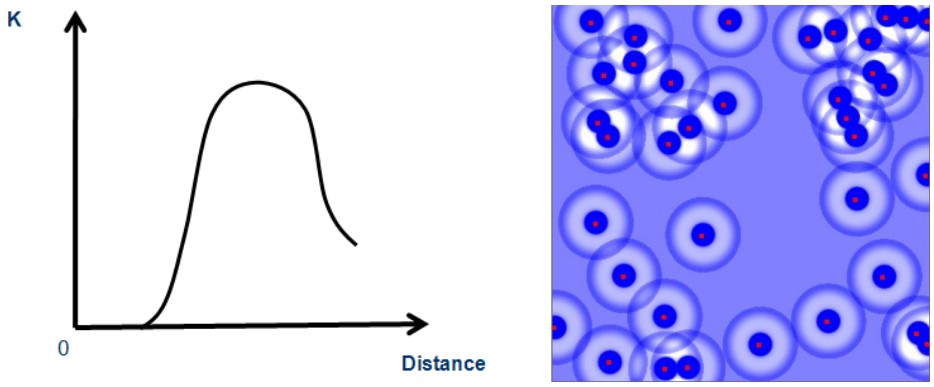


(d) Eventually Poisson-disk pattern is obtained.

Figure 5.8: An example illustrating how point distributions are obtained using the approach presented by Lane [54], and with the kernel function of Figure 5.7. Left: the point distribution. Right: the probability density function. Dark blue disks represent regions where probability is zero.

Figure 5.9 shows a kernel function, and the corresponding point distribution. The

kernel function sets the probability density in the immediate negihbourhood N of the generated point to zero, but increases it in the region beyond N, thus increasing the probability that the points will be clustered together, as shown in Figure 5.9b.



(a) (b)

Figure 5.9: (a) A kernel function; (b) The corresponding point distribution and the probability density field obtained using Lane's method [54].

5.3 Summary

In this chapter, an overview of existing methods for generating point set distributions was presented. The point sets, in this thesis, are used for generating trees, and for growing structures in proximity to surfaces. The next chapter describes the methods that were used to generate point distributions. These distributions are generated in the space enclosed by implicit surfaces, and close to the boundary of the surfaces.

Chapter 6

Point Set Generation — Implementation

In this chapter, details of the algorithms that were used to generate point set distributions are provided. These distributions are used for two purposes. First, points can be generated inside the space enclosed by implicit surfaces. These points are used by the space colonization algorithm to generate trees. By changing distributions, variations in trees can be introduced. Further, the point set can be limited to the proximity of surfaces, which is useful for growing trees around these surfaces. The details are presented in the following sections.

6.1 Generating points inside an implicit surface

For each implicit surface, the implicit modeler (Chapter 4) creates an axis-aligned bounding box. An implicit surface lies inside its bounding box.

The importance of the bounding box stems from the fact that the point set generation algorithms listed in this chapter use these to generate points in the volume enclosed by the implicit surfaces. First, a random point is generated from a uniform distribution inside the bounding box of a surface. Then a test is made to check if the point lies inside the implicit surface. If the point lies outside the surface, then the point is rejected and the process continues until a point inside the implicit surface is generated. An axis-aligned bounding box is used since the random number generator creates each component of a random point in $[0, 1]$, and this point can be easily scaled

and translated into the bounding box.

In this thesis, Lane's distribution method [54, 55] was used for generating point distributions. This method was discussed in Section 5.2.2 for 1D and 2D distributions, and the details are the same for generating 3D distributions. This method generates a point inside a 3D grid, and this point is transformed to the bounding box of the implicit surface.

The advantage of using Lane's method for generating point sets is that different distributions can be generated by specifying an appropriate kernel function. The underlying implementation remains unchanged. Figure 6.1 shows the cross-section of a probability density function obtained using the kernel function of Figure 5.7. In the figure, the dark blue disks represent the regions where the probability is zero. The disks are of different radii since the figure shows the cross-section of a 3D probability density function. Figure 6.2 shows the cross-section of a probability density function obtained using the kernel function of Figure 5.9.

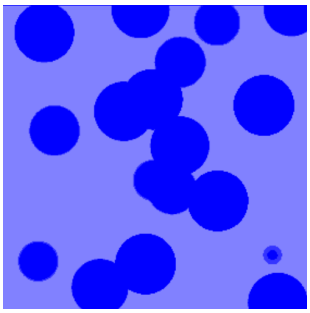


Figure 6.1: Cross-section of a probability density function obtained using Lane's distribution with the kernel function of Figure 5.7. The dark blue disks represent regions with zero probability. Since the figure shows a cross-section of a 3D density function, the blue disks are of varying radii.

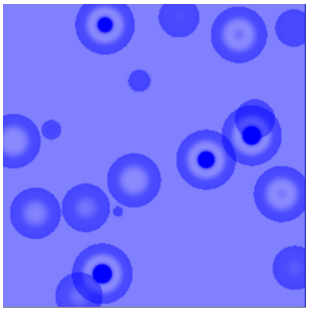


Figure 6.2: Cross-section of a probability density function obtained using Lane's distribution with kernel function of Figure 5.9.

Lane's method draws samples from a given probability density function, and hence the point distributions can be further controlled by using appropriate probability density functions. In particular, the density functions can depend upon the field values of implicit surfaces. Figure 6.3 shows two examples. In Figure a, the density function is non-zero only for the narrow region close to the boundary of the surface, and hence the resulting points are generated close to the boundary. In Figure b, the probability density function is inversly proportional to the field values. As a result, the resulting distribution is dense near the boundary, while it is sparse near the center of the circle.

6.2 Generating points in proximity to surfaces

This section presents algorithms to generate points in proximity to surfaces. These point sets were used to grow structures around surfaces.

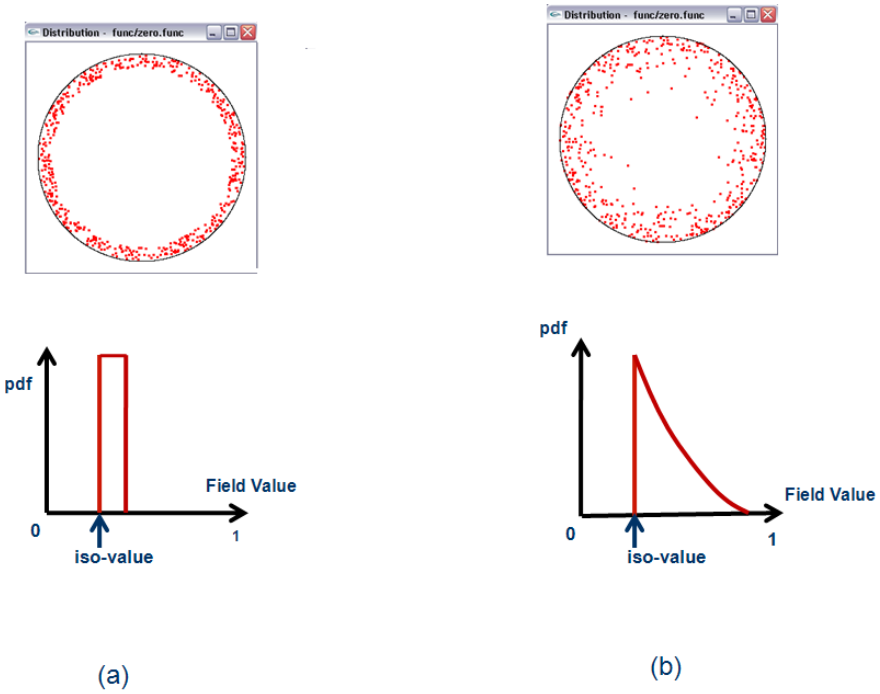


Figure 6.3: Controlling point distributions using field values of implicit surfaces.

6.2.1 Generating points close to implicit surfaces

The methods described in Section 6.1 can also be used for generating points outside and in proximity to implicit surfaces. A uniform number generator generates a random point inside a unit cube. This point is scaled and translated inside the axis-aligned bounding box of the implicit surface. The bounding box is defined such that it encloses all points whose field value is not zero. The random point is accepted if it lies close to the boundary of the surface, otherwise it is rejected. Suppose f is a given implicit function. Given *iso-value* and a point p, the point is outside and in proximity to the implicit surface if it satisfies the following condition:

$$f(p) \in (\text{isov} - \epsilon, \text{isov})$$

where $\epsilon > 0$ is a user-specified parameter.

This approach of using the rejection criterion for generating point distributions is very ineffecient, since the volume of the region in proximity to a surface where points are to be generated is small compared to the region inside the surface. This would lead to a large number of samples being generated, and rejected. If the field function is complex, the rejection cost would be very high. A possible improvement would be the selection of a bounding box which is not axis-aligned, and encloses the implicit surface more tightly than the axis-aligned bounding box. This would, however, not solve the problem of a large available region inside the surface, and the number of samples rejected would still be very high.

To avoid slow generation times, two optimizations were implemented that are described in the following sections.

6.2.2 Caching field values

The objective of this optimization is to reduce the time it takes to evaluate a field function. Thus, even though a large number of samples are generated and rejected, the overall computation time is reduced.

The strategy used is to cache field values for regions that contain points in proximity to surfaces, and by identifying if a point belongs to the proximity of the surface (point classification). Both of these objectives are achieved by marching cubes around a surface using the approach described by Wyvill et al. [103]. The cubes thus created are stored in a hash table. The field values at the corners of cubes are also cached (as was done by Schmidt [85].) If for a point p, there exists a cube in the hash table that encloses the point, then the point is a potential candidate of the

Without cache	94.7 seconds
With cache	4.1 seconds

Table 6.1: Time of generating 1000 points, with and without cache, for a surface in Figure 6.4. For building the cache, a cube of edge length 0.5 is used. Total number of cubes in the cache is 2275. The points are generated between the field value 0.40 and 0.41.

point distribution, otherwise it is rejected. The field value at the point is approximated by tri-linearly interpolating the field values at the eight corners of the cube that encloses the point. The computation time for a rejected sample is equivalent to a hash table lookup.

Table 6.1 shows time of generating 1000 points, with and without cache, for a surface in Figure 6.4. The second method – where the cache was used – is about twenty three times faster than the first method. However, in a simple case where the field value computation is fast (such as for a sphere), this method may actually increase the overall time to generate points. For instance, generating 1000 Poisson-disk points using the cache for a sphere of radius 10 was about twenty times slower than generating the points directly.

As seen in the examples above, the use of the cache reduces the computation time only if the hash table look up and tri-linear interpolation of field values is faster than the computation of field values, which is typically the case if the Blob tree [102] has a large depth, and the speed up by caching occurs by not explicitly traversing the Blob tree.

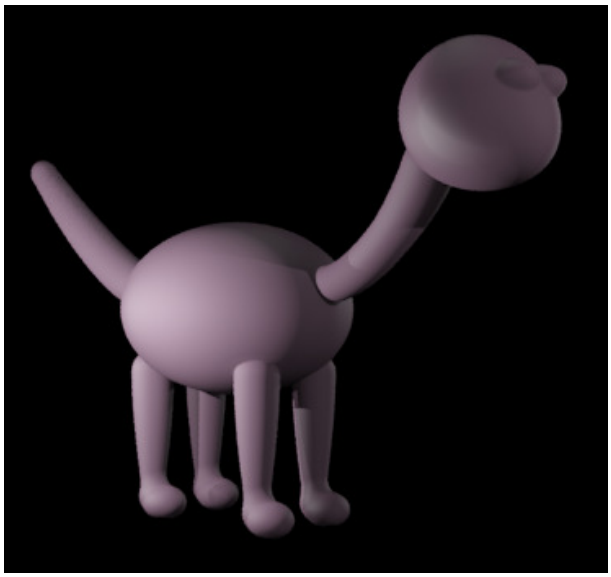


Figure 6.4: Test surface for profiling point set generation time

6.2.3 Optimizing using Lane's distribution method

The cache (previous section) tries to optimize point set generation by reducing the field value computation time. A large number of samples, however, are rejected. In the second optimization, the objective is to reduce the number of samples that are rejected by making use of the Lane's distribution method [54, 55]. This method, as discussed in Section 5.2.2, draws samples from a given probability density function. Thus, if the domain of the probability density function is the same as the bounding box of the implicit surface, and the probability density function is non-zero only for regions close to the boundary of the surface, then all the samples would be generated in proximity to the surface. The key to initializing the probability density function (pdf) is the observation that in Lane's distribution, the domain of the pdf is divided into voxels of equal size, and the function is constant in each voxel. Hence, the pdf for a voxel is initialized to a non-zero value if the center of the voxel is in proximity

to the implicit surface, otherwise the pdf is set to zero.

In Figure 6.5, points are generated using this method. Note that no point is generated inside the circle and all the generated points are close to the boundary of the circle.

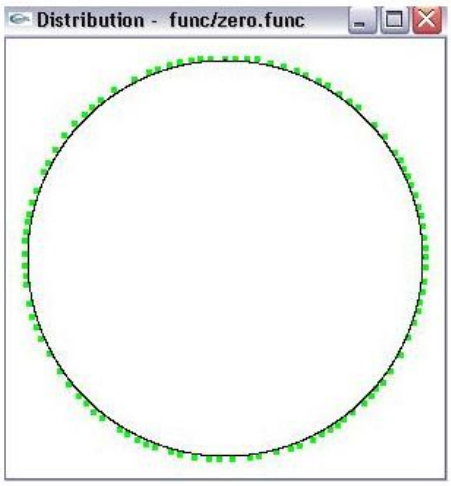


Figure 6.5: Restricting points outside and in proximity to the circle using Lane's distribution method [54, 55].

This method, however, adds the overhead of initializing the pdf – we need to evaluate the field value at the center of each voxel to determine if the voxel lies inside, or outside and close to the surface and then initialize it appropriately. We can, however, optimize by defining the field value approximation, as described in the previous section. A point that would be rejected is assigned a probability of 0, otherwise it is assigned a probability of 1.

Table 6.2 shows the time taken, using these approaches, to generate 1000 Poisson-disk points for the implicit surface in Figure 6.4.

Lane's distribution method	145.9 seconds
Initialize grid without cache	45.01 seconds
Initialize grid with a cache	8.67 seconds

Table 6.2: Time taken to generate 1000 Poisson-disk points, using Lane's distribution, on a surface of Figure 6.4. For building the cache, a cube of edge length 0.5 was used. Total number of cubes in the cache is 2275. The points are generated between the field value 0.40 and 0.41.

6.3 Generating points close to triangular meshes

The methods described above work for implicit surfaces. However, many models are defined in a polygon format [1, 2, 3]. The points are generated in proximity to these meshes in order to grow trees around these meshes.

If an implicit function can be created from a given polygon mesh, then the methods discussed in Section 6.2 can be used to generate points close to implicit surfaces. There has been work done in this direction. For instance, Yngve et al [105] use variational methods [93] (Section 3.2) to create an implicit function from a given mesh. This method, however, is computationally very expensive. Hence, a different approach is used as outlined below.

It is assumed that the mesh is oriented 2-manifold. A triangle mesh consists of a list of vertices and a list of faces (triangles) where each face consists of three indices into the vertex list. To generate a candidate point q, first a face is randomly selected, and then a point p is randomly selected on this face. A random point p can be generated inside a triangle using the method presented by Turk [92]: Two uniform random variables s and t are drawn such that $s+t<1$. If v_1, v_2, and v_3 are the vertices of the triangle, then $p=(1-s-t)\,v_1+s\,v_2+t\,v_3$. If n is the interpolated

face normal (as in Phong shading), then the candidate point q is translated along the face normal: $q = p + u * n$ where u is a random number between the user-specified limits.

Using this approach, the triangles of a mesh are selected from a uniform distribution. If a mesh consists of a large number of small triangles and relatively small number of large triangles, then this method would generate points that would not cover the surface well. Figure 6.6a shows an example. The density of points is high at the top and bottom, where presumably the triangles are small, while it is sparse in the center.

To address this problem, triangles of a mesh can be selected from a probability distribution that is proportional to the area of the triangle. The inverse cumulative distribution method, as discussed in Section 5.2.2, can be used for this purpose. Details are as follows.

Suppose a mesh has N triangles. Let A_i be the area of the i_{th} triangle where i is a 0-based index. Define a probability density function $p(x)$ over interval $[0, N]$ such that for $x \in [i, i+1)$, $p(x)$ is proportional to the area of the i_{th} triangle. Specifically

$$p(x) = \frac{A_i}{\sum_{j=0}^{N-1} A_j} \qquad (x \in [i, i+1))$$

Then, cumulative distribution function $F(x)$ is defined as

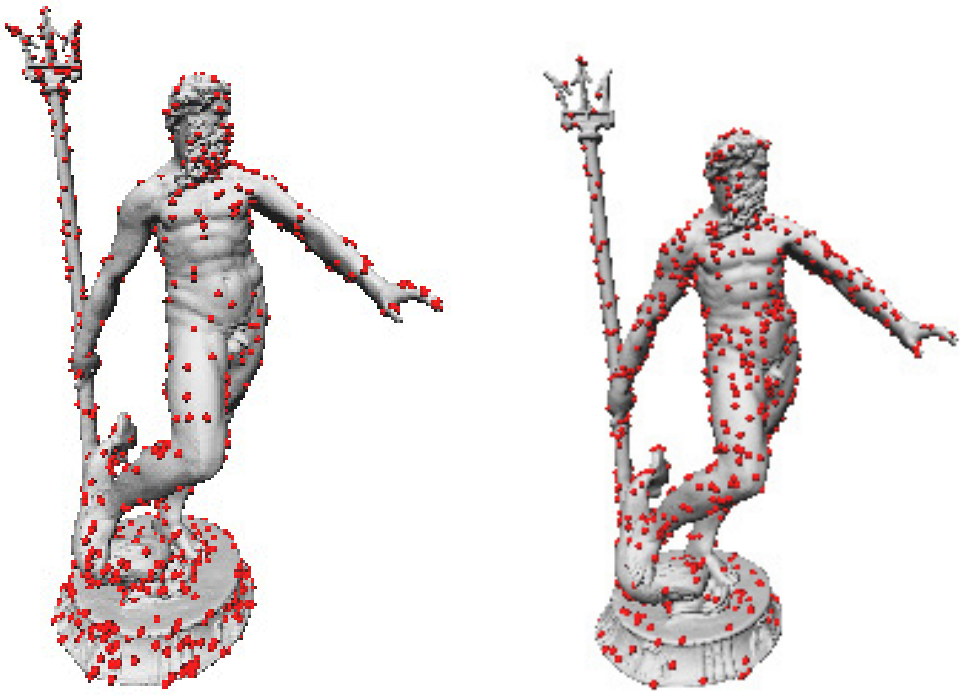
$$F(x) = \int_0^x p(x)\, dx = \frac{\sum_{j=0}^{i-1} A_j + A_i * (x - i)}{\sum_{j=0}^{N-1} A_j}$$

The cumulative distribution function is computed for discrete points ($i = 0, 1, 2, ...$) and saved in a *cdf* array. The cdf array is used in equation 6.1 below.

Now to select the triangle *index*, we first draw a uniform random variable $u \in (0, 1)$ and find $F^{-1}(u)$, that satisfies the following relation

$$\text{cdf[index]} <= u < \text{cdf[index+1]} \tag{6.1}$$

Figure 6.6b shows points distributed using this approach. As shown in Figure b, the points cover the mesh more uniformally than the points in Figure a.



(a) (b)

Figure 6.6: Generating points close to a mesh using algorithm that selects triangles from a uniform distribution and algorithm that selects triangles from a probability distribution that is proportional to triangle area. Model is provided courtesy of IMATI/INRIA by the AIM@SHAPE Shape Repository

6.4 Summary

In this chapter, algorithms for generating point set distributions were discussed. Point sets are generated within the region enclosed by an implicit surface. These points can then be used by space colonization algorithms for generating tree structures. Further, algorithms for generating point sets close to implicit surfaces and triangular meshes were discussed.

Chapter 7

Modeling Trees

This chapter provides a description of the working of all the three components of the proposed system — interactive definition of the tree crowns, populating these crowns with points, and generating trees. The tree architecture can be influenced by the specification of crown shapes and by varying point distributions. A particularly interesting case occurs when we restrict point sets outside and in proximity to surfaces. With these point sets, trees grow around surfaces. Furthermore, the resulting tree structures suggest the shapes of the underlying surfaces.

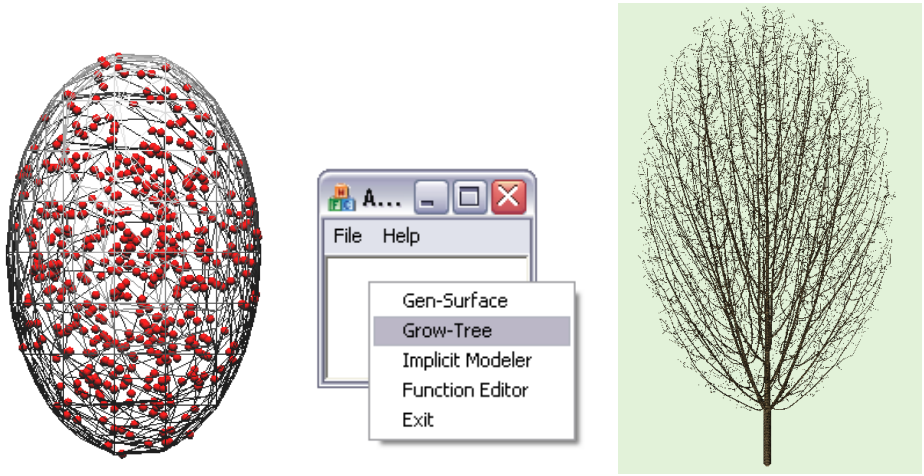
7.0.1 The Modeling process

To generate a tree, a tree crown is first defined using the implicit modeler. As mentioned in Chapter 4, the user can interact with the modeler in various ways. One of the methods is using the textual commands. These commands can be saved to a file. This is especially useful if the user later wishes to modify parameters or regenerate the model. For instance, the user can give the following commands to define a crown, generate Poisson-disk point distribution in the crown and save the points to a text file. The properties of the Poisson-disk point distribution — such as the minimum distance between points and the number of points — is specified in a file (*poisson.props* in the listing below):

```
sphere s
scale s 50 80 50
```

```
point_set_gen s poisson poisson.props
save_point_set points.txt
```

The result of executing these commands is shown in Figure 7.1. The figure shows the interface of an application that was motivated by the vlab object manager [63], and was written to facilitate interaction between various components. The application allows users to specify the menu (shown in Figure b) in a text file, and associate actions with each menu item. The allowable actions are setting environment variables, changing current directories (on Windows), and running other applications. The objective was to minimize the number of steps the user has to take to generate a tree model. Figure c shows the tree model generated using the previous commands.

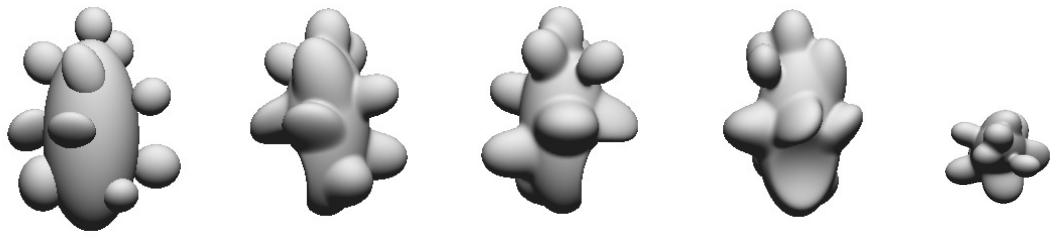


(a) (b) (c)

Figure 7.1: Tree generation process. a) A crown defined using the implicit modeler, and populated with points; b) An application to communicate with various components of the system; c) The resulting tree.

Figures 7.2, 7.3, and 7.4 show the relationship between crown envelope and the resulting tree, shown in different views. A tree such as this could not be generated in the original formulation.

The branching structures of trees can be further influenced by varying point distributions, as shown in Figure 7.5. In Figure a, Poisson-disk distribution is used, and the resulting crown uniformally fills the space. In Figure b, points are generated close to the boundary of the surface (using the approach of Section 6.1, Figure 6.3a), and this results in a tree whose branches tend to grow towards the boundary of the surface. The tree of Figure c is generated using a clustered distribution, and this results in masses of branches concentrated in small regions. The tree in Figure d was created using the method described in Section 6.1, Figure 6.3b. The density of points is inversly proportional to the field values of the implicit surfaces.



(a) (b) (c) (d) (e)

Figure 7.2: a) Implicit surfaces used for defining the crown envelope, and the resulting crown in various views; b) Front view; c) Back view; d) Side view; e) Top view.

(a)

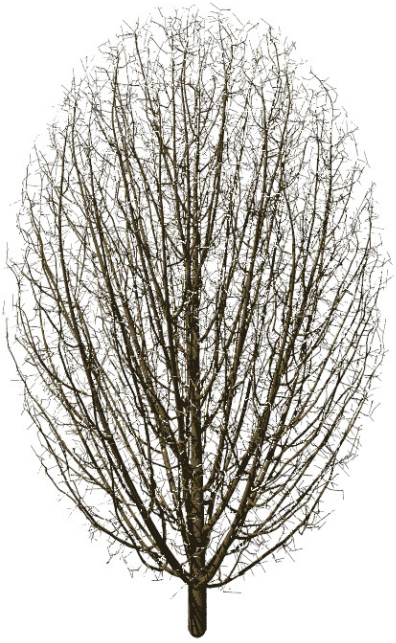
(b)

Figure 7.3: Front and back views of the tree for the crown envelope of Figure 7.2.

(a)

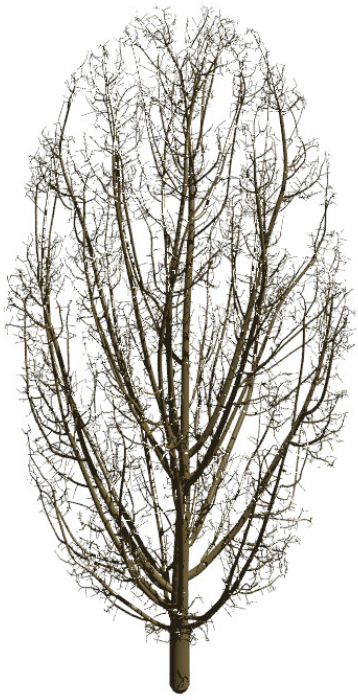
(b)

Figure 7.4: Side views of the tree for the crown envelope of Figure 7.2.



(a) Poisson distribution

(b) Poisson distribution along the crown boundary



(c) Clustered distribution

(d) Point distribution based on field values, as shown in Figure 6.3b.

Figure 7.5: Generating plants by varying point distributions.

7.1 Interactive manipulation of point distributions

The previous chapter described methods to generate point sets with different distributions. The user can further manipulate these point sets using two methods. First, the user can reduce the density of points in a given region of the crown by using a secondary implicit surface. For instance, given an implicit surface named *crown* and a secondary implicit surface named *secondary*, that occupies some region of the crown, the following command can be specified to reduce by eighty percent the density of points in the region enclosed by *secondary*:

```
point_set_thin crown secondary 80
```

The operation works as follows. Let n be the number of points that are inside the secondary implicit surface, and d be a user-specified parameter indicating the factor by which the density of points should be reduced. Then, the first $\frac{d}{100}n$ points are removed from the distribution. Implicit surfaces make it very easy to perform this operation, since points can be easily classified as inside, or outside the surface.

Secondly, the distribution of a given region of a crown can be made more dense by using a secondary implicit surface that encloses the given region, and generating points in this surface.

Figure 7.6 shows models of two trees generated using a spherical crown. In the second figure, the density of points was reduced in a region using a secondary implicit surface.

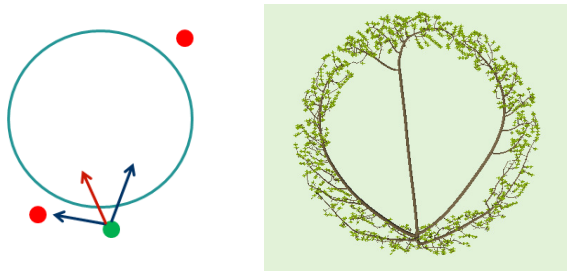
(a) (b)

Figure 7.6: Manipulating tree shapes by interactively reducing the density of points in some regions of the crown.

7.2 Growing trees around surfaces

To grow trees that surround a given surface, do not pass into or wander away from the surface, it seems logical to generate point sets outside but in proximity to the surface. If marker points are inside the space enclosed by the surface, they will attract the branches into the surface. We can, however, come up with scenarios where trees enter the interior of the surface even when points are limited to the proximity of the surface, as Figure 7.7 illustrates. The figure shows a cross-section of a sphere and a tree node (green circle). The tree node is influenced (blue arrows)

by two marker points (red circles). The resultant direction (red arrow), where a new node will be created, points inside the surface. Figure 7.7b shows branches growing away from the boundary of the torus. This is due to the fact that the marker points toward the north end influence the tree nodes at the south end.

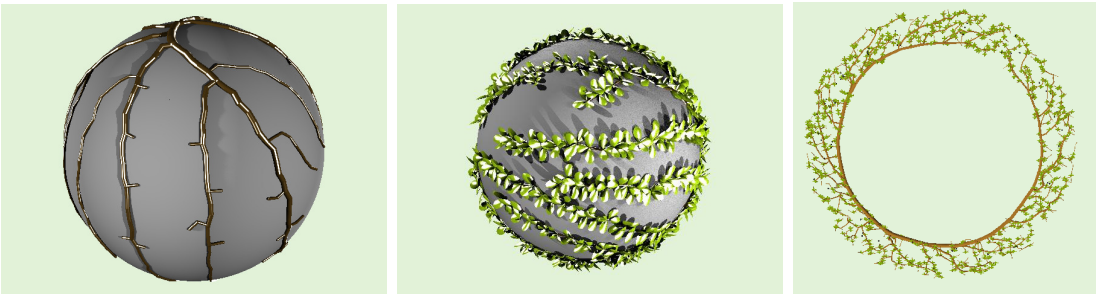


(a) (b)

Figure 7.7: Examples of cases when the branching structure may enter into the surface or grow away from the boundary of the surface. a) Cross-section of a sphere with marker points (red circles) and a tree node (green circle). The direction where a new tree node will be created points inside the surface (red arrow); b) A branching structure grows away from the boundary of the torus.

These examples suggest that not all marker points should influence the nearest tree node. By using a small radius of influence, the tree nodes are not attracted by marker points that are far way from them, thus avoiding the problems illustrated in Figure 7.7. Figure 7.8 shows the structures grown around spheres and a torus, by limiting points outside and in proximity to these surfaces, and by using a small radius of influence.

The figures in the following pages illustrate examples of structures generated in proximity to implicit surfaces and triangular meshes (Figures 7.9 and 7.10). It is interesting to compare Figure 7.11 with the topiary dino of [75]. In [75], trees were grown within the volume of a predefined surface. The branches that grew outside



(a) (b) (c)

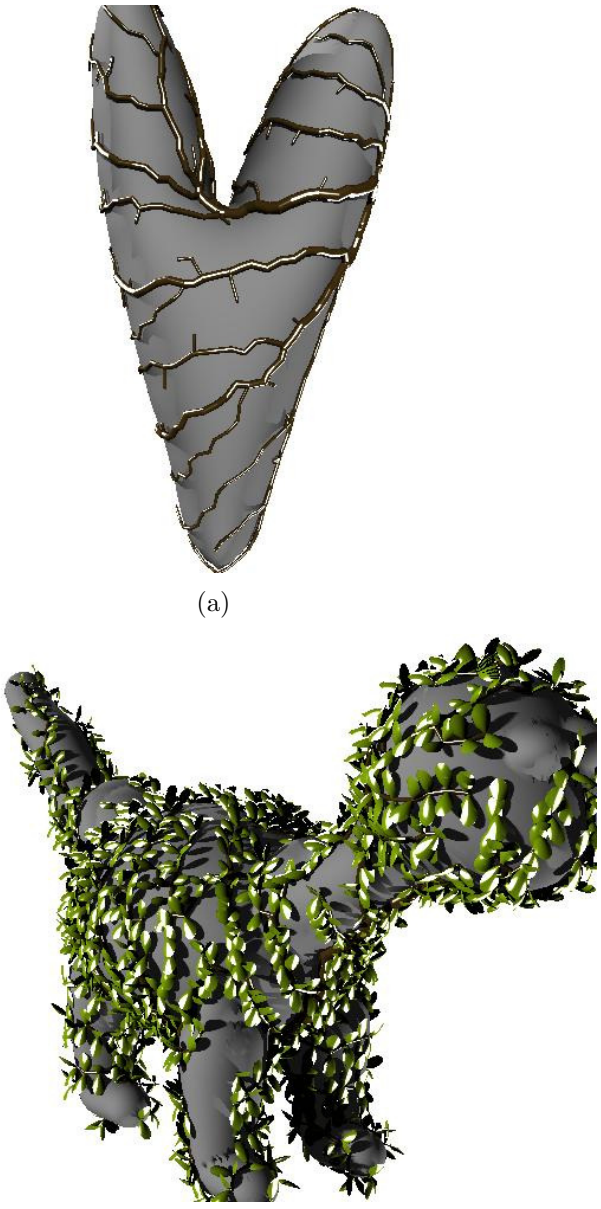
Figure 7.8: Growing trees around spheres and a torus by limiting point distributions outside and in proximity to surfaces, and by using a small radius of influence.

the surface were cut off. This promoted outgrowth of lateral branches. Figure 7.11 achieves a similar visual effect by growing structures in proximity to surfaces.

Figures 7.12 and 7.13 illustrate that the grown structures suggest the shape of the underlying surfaces, and can be used for decoration and ornamental patterns.

7.3 Summary

This chapter described the working of all the three stages of the proposed system — interactive specification of tree crowns, generation of points in these crowns, and generating trees using the space colonization algorithm. It was shown that the shape and branching strucutre of the tree can be manipulated by the specification of crown shapes and by varying the point distributions. Furthermore, trees can be grown around surfaces by restricting point sets outside and in proximity to surfaces. The shape of the surface is suggested by the branching structure that surrounds it.



(a)

(b)

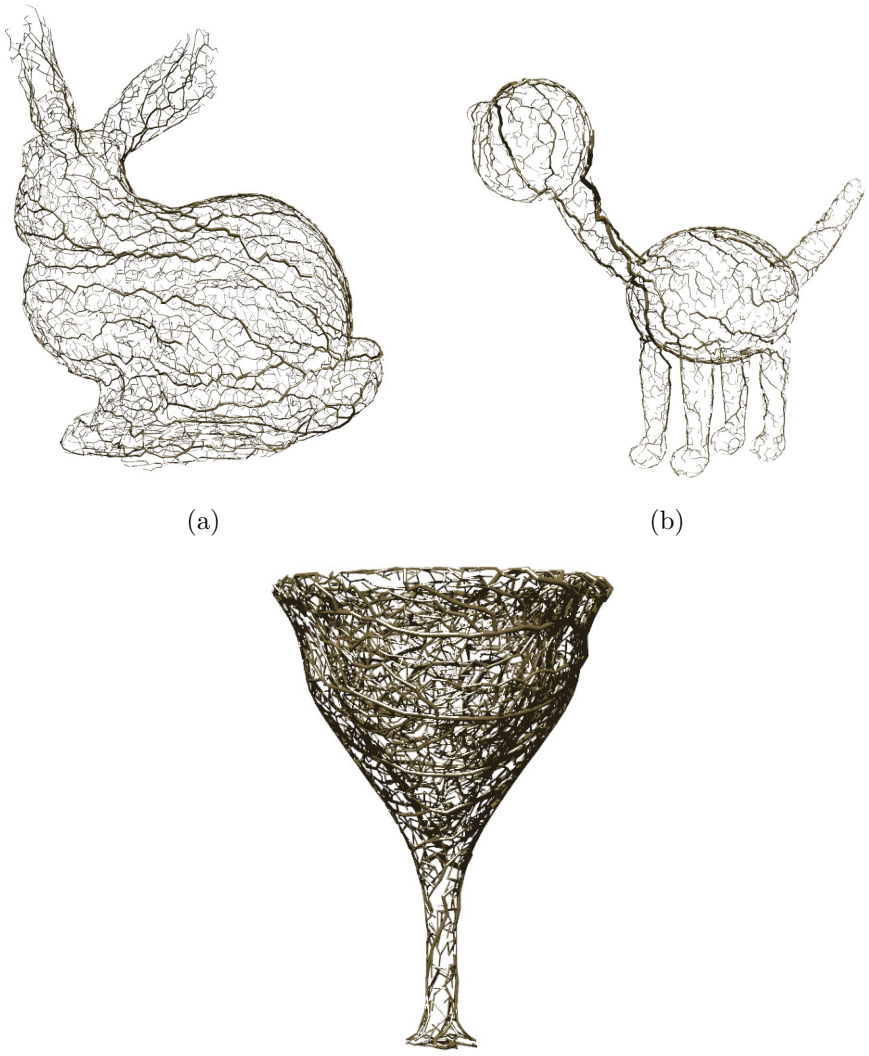
Figure 7.9: Growing structures around implicit surfaces.

Figure 7.10: Growing a structure in proximity to a triangular mesh. The mesh is a courtesy of Aim@Shape IMATI/INRIA.

(a)

(b)

Figure 7.11: Achieving the visual effects of topiary. In both the figures, the surfaces used are implicit surfaces (Chapter 4).



(a) (b)

(c)

Figure 7.12: Trees suggesting the shapes of the underlying surfaces. The underlying surface used for a) is the Stanford bunny, while implicit surfaces (Chapter 4) were used for b) and c).

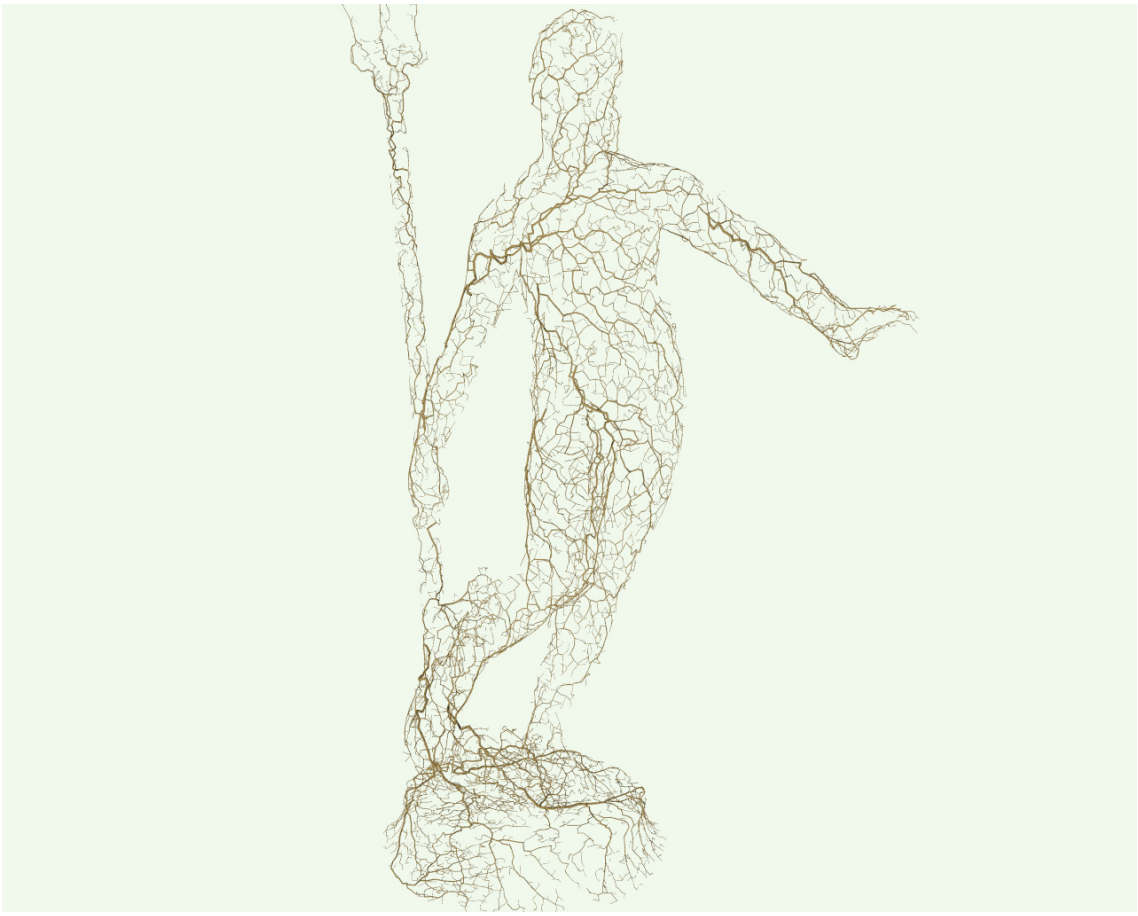


Figure 7.13: Growing a structure in proximity to a triangular mesh. The underlying mesh, courtesy of Aim@Shape IMATI/INRIA, is not shown. The grown structure suggests the shape of the underlying mesh (see Figure 6.6).

Chapter 8

Conclusions

In this work, implicit surfaces are used for the modeling of plant structures. There are three components to the proposed system. First, an implicit modeling system is implemented. The implicit modeler is used to interactively specify the shapes of the tree crowns. The implicit modeler incorporates features such as distance surfaces, implicit surfaces of revolution, variational surfaces, sweep surfaces, blending, CSG, deformations, and implicit skinning. The user interacts with the system using a graphical interface and a command editor, which can be used to specify textual commands to define and manipulate the models.

The implicit skinning framework was discussed in detail. Skinning, also known as lofting, is a popular modeling paradigm in the parametric domain, but it hasn't caught up in the implicit arena. Existing approaches construct expensive implicit functions from parallel contours [84, 17, 4, 93]. Further, these approaches loft cross-sections along straight line segments. The method presented here uses interpolation of distance fields. It is shown that the field values produced by this process are smooth and bounded and hence the resulting surfaces can be further modified by the summation blending. Another benefit of using this approach is that it seamlessly handles complex cases such as those shown in Figures 4.7 and 4.10.

The second part of the proposed system involves generating point set distributions in the space enclosed by implicit surfaces, or in proximity to surfaces. The density of these point distributions is further controlled by the field values of the implicit

surfaces.

Points are also generated close to triangular meshes. A method was presented that generates a point by first selecting a triangle of the mesh from a uniform distribution, and then generating a point in proximity to the selected triangle. This method, however, poorly samples those triangular meshes which consist of triangles with greatly varying areas; in particular, triangular meshes that have a large number of small triangles and relatively small number of larger triangles (Figure 6.6). To address this problem, a method was presented that selected triangles from a probability distribution that is proportional to the area of triangles.

The point distributions are interactively manipulated using implicit surfaces. Specifically, the density of points in any given region of the crown is reduced by introducing a secondary implicit surface in the desired region, and deleting user-specified percentage of points in the volume enclosed by the secondary implicit surface.

The proposed system extends the original space colonization algorithm in several directions [83]. The use of implicitly defined surfaces as crown envelopes extends the range of crown shapes that can be defined, thus improving the flexibility of the space colonization algorithm. This flexibility is further increased by the possibility of defining different probability distributions for placing the markers of free space, either in the entire volume enclosed by the surface, or close to the surface. Further, growing a tree near a surface offers an unexpected visual quality, allowing a shape to be suggested by the branching structure that surrounds it. This may be used for illustrative and NPR purposes.

For a future work, the use of implicit skinning for defining the crown shapes could be explored. By using this approach, a wide variety of crowns can be easily defined by interactively sketching the cross-section curves.

Bibliography

[1] *AIM@SHAPE Shape Repository v4.0*. http://shapes.aimatshape.net/.

[2] *Large Geometric Models Archive*. http://www.cc.gatech.edu/projects/large_models/.

[3] *The Stanford 3D Scanning Repository*. http://graphics.stanford.edu/data/3Dscanrep/.

[4] Samir Akkouche and Eric Galin. Implicit surface reconstruction from contours. *Vis. Comput.*, 20(6):392–401, 2004.

[5] M. T. Allen, P. Prusinkiewicz, and T. M. Dejong. Using L-systems for modeling source-sink interactions, architecture and physiology of growing trees: the L-PEACH model. *New Phytologist*, 166(3):869–880, June 2005.

[6] Fabricio Anastacio, Mario Costa Sousa, Faramarz Samavati, and Joaquim A. Jorge. Modeling plant structure using concept sketches. In *NPAR '06: proceedings of the 4th international symposium on Non-photorealistic animation and rendering*, pages 105–113. ACM, 2006.

[7] M. Aono and T.L. Kunii. Botanical tree image generation. *IEEE Computer Graphics and Applications*, pages 10–34, 1984.

[8] James Arvo, David Kirk, and Apollo Computer. Modeling plants with environment-sensitive automata. pages 27–33, Melbourne, Australia, 1988. Proceedings of Ausgraph.

[9] Michael Balzer, Thomas Schlömer, and Oliver Deussen. Capacity-constrained

point distributions: a variant of lloyd's method. *ACM Trans. Graph.*, 28(3):1–8, 2009.

[10] Bedrich Benes and Erik Uriel Millan. Virtual climbing plants competing for space. pages 33–42. IEEE Computer Society, 2002.

[11] James F. Blinn. A generalization of algebraic surface drawing. *ACM Trans. Graph.*, 1(3):235–256, 1982.

[12] Jules Bloomenthal. Modeling the mighty maple. In *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pages 305–311, New York, NY, USA, 1985. ACM.

[13] Jules Bloomenthal. Calculation of reference frames along a space curve. *Graphics gems*, pages 567–571, 1990.

[14] Jules Bloomenthal. An implicit surface polygonizer. In *Graphics Gems IV*, pages 324–349. Academic Press, 1994.

[15] Jules Bloomenthal and Brian Wyvill, editors. *Introduction to Implicit Surfaces*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.

[16] R. Borchert and H. Honda. Control of development in the bifurcating branch system of tabebuia rosea: A computer simulation. In *Botanical Gazette 145, 2*, pages 184–195, 1984.

[17] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3d objects with radial basis functions. In *SIGGRAPH '01: Proceedings of the 28th annual*

conference on Computer graphics and interactive techniques, pages 67–76, New York, NY, USA, 2001. ACM.

[18] Norishige Chiba, Ken Ohshida, Kazunobu Muraoka, Mamoru Miura, and Nobuji Saito. A growth model having the abilities of growth-regulations for simulating visual nature of botanical trees. *Computers & Graphics*, 18(4):469–479, 1994.

[19] S. Muraoka-K. Miura M. Chiba, N. Ohkawa. Visual simulation of botanical trees based on virtual heliotropism and dormancy break. In *Journal of Visualization and Computer Animation*, pages 313–322, Great Britain, 1994. John Wiley and Sons Ltd.

[20] Michael F. Cohen, Jonathan Shade, Stefan Hiller, and Oliver Deussen. Wang tiles for image and texture generation. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 287–294, New York, NY, USA, 2003. ACM.

[21] Robert L. Cook. Stochastic sampling in computer graphics. *ACM Trans. Graph.*, 5(1):51–72, 1986.

[22] Benoit Crespin, Carole Blanc, and Christophe Schlick. Implicit sweep objects. *Computer Graphics Forum*, 15:165–174, 1996.

[23] Phillippe de Reffye, Claude Edelin, Jean Françon, Marc Jaeger, and Claude Puech. Plant models faithful to botanical structure and developmentr. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 151–158, New York, NY, USA, 1988. ACM.

[24] Oliver Deussen, Pat Hanrahan, Bernd Lintermann, Radomír Měch, Matt Pharr, and Przemyslaw Prusinkiewicz. Realistic modeling and rendering of plant ecosystems. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 275–286, New York, NY, USA, 1998. ACM.

[25] Oliver Deussen, Stefan Hiller, Cornelius van Overveld, and Thomas Strothotte. Floating points: A method for computing stipple drawings. *Computer Graphics Forum*, 19:40–51, 2000.

[26] Oliver Deussen and Bernd Lintermann. *Digital Design of Nature, Computer Generated Plants and Organics*. Springer, 2004.

[27] Mark A. Z. Dippé and Erling Henry Wold. Antialiasing through stochastic sampling. *SIGGRAPH Comput. Graph.*, 19(3):69–78, 1985.

[28] Daniel Dunbar and Greg Humphreys. A spatial data structure for fast poisson-disk sample generation. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, pages 503–508, New York, NY, USA, 2006. ACM.

[29] Philip Dutre, Kavita Bala, and Philippe Bekaert. *Advanced Global Illumination*. A. K. Peters, Ltd., Natick, MA, USA, 2002.

[30] R. W. Floyd and L. Steinberg. An adaptive algorithm for spatial gray scale. In *SID, Int. Symp. Dig. Tech*, pages 36–37, 1975.

[31] D. Frijters. Mechanisms of developmental integration of aster novae-angliae l., and hieracium murorum l. pages 561–575. Annals of Botany 42, 1978.

[32] D. Frijters. Principles of simulation of inflorescence development. pages 549–560. Annals of Botany 42, 1978.

[33] D. Frijters and A. Lindenmayer. A model for the growth and flowering of aster novae-angliae on the basis of table (1, 0) lsystems. pages 24–52. Springer-Verlag, Berlin, 1974. Lecture Notes in Computer Science 15.

[34] D. Frijters and A. Lindenmayer. Developmental descriptions of branching patterns with paracladlal relationships. pages 57–73. In A. Lindenmayer and G. Rozenberg (Eds.): Automata, languages, development, North-Holland, Amsterdam, 1976.

[35] Callum Galbraith, Peter MacMurchy, and Brian Wyvill. Blobtree trees. In *CGI '04: Proceedings of the Computer Graphics International*, pages 78–85, Washington, DC, USA, 2004. IEEE Computer Society.

[36] N. Greene. Voxel space automata: modeling with stochastic growth processes in voxel space. In *SIGGRAPH '89: Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, pages 175–184, New York, NY, USA, 1989. ACM.

[37] C. Grimm. Implicit generalized cylinders using profile curves. *Implicit Surfaces 99*, pages 33–41, 1999.

[38] John C. Hart. Sphere tracing: a geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer*, 12:527–545, 1996.

[39] Gabor T. Herman, Jingsheng Zheng, and Carolyn A. Bucholtz. Shape-based interpolation. *IEEE Comput. Graph. Appl.*, 12(3):69–79, 1992.

[40] Stefan Hiller, Oliver Deussen, and Alexander Keller. Tiled blue noise samples. In *VMV '01: Proceedings of the Vision Modeling and Visualization Conference 2001*, pages 265–272. Aka GmbH, 2001.

[41] Wu F.-C. Chen B.-Y. Chuang Y.-Y. Ho, C.-C. and M. Ouhyoung. Cubical marching squares: Adaptive feature preserving surface extraction from volume data. In *Computer Graphics Forum*, pages 537–545, 2005.

[42] Matthew Holton. Strands, gravity and botanical tree imagery. *Computer Graphics Forum*, 13(1):57–67, March 1994.

[43] 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. *Journal of Theoretical Biology*, 31:331–338, 1971.

[44] Takashi Ijiri, Shigeru Owada, and Takeo Igarashi. Seamless integration of initial sketching and subsequent detail editing in flower modeling. In *Eurographics*, pages 617–624, Vienna, Austria, September 2006.

[45] Takashi Ijiri, Shigeru Owada, and Takeo Igarashi. The sketch L-System: Global control of tree modeling using free-form strokes. In *Smart Graphics*, pages 138–146, 2006.

[46] L M. Janssen and A. Lindenmayer. Models for the control of branch positions

and flowering sequences of capitula in mycelis muralis (l.) dumont (compositae). pages 191–220. New Phytologist 10J, 1987.

[47] Thouis R. Jones. Efficient generation of poisson-disk sampling patterns. *Journal of Graphics Tools*, 11(2):27–36, 2006.

[48] Tao Ju, Frank Losasso, Scott Schaefer, and Joe Warren. Dual contouring of hermite data. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 339–346, New York, NY, USA, 2002. ACM.

[49] Leif P. Kobbelt, Mario Botsch, Ulrich Schwanecke, and Hans-Peter Seidel. Feature sensitive surface extraction from volume data. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 57–66, New York, NY, USA, 2001. ACM.

[50] Johannes Kopf, Daniel Cohen-Or, Oliver Deussen, and Dani Lischinski. Recursive wang tiles for real-time blue noise. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, pages 509–518, New York, NY, USA, 2006. ACM.

[51] Ares Lagae and Philip Dutré. A procedural object distribution function. *ACM Trans. Graph.*, 24(4):1442–1461, 2005.

[52] Ares Lagae and Philip Dutré. An alternative for wang tiles: colored edges versus colored corners. *ACM Trans. Graph.*, 25(4):1442–1459, 2006.

[53] Ares Lagae and Philip Dutré. A comparison of methods for generating Poisson disk distributions. Report CW 459, Department of Computer Science,

K.U.Leuven, Leuven, Belgium, August 2006.

[54] Brendan Lane. Models of plant communities for image synthesis. Master's thesis, Department of Computer Science, University Of Calgary, Alberta, June 2002.

[55] Brendan Lane and Przemyslaw Prusinkiewicz. Generating spatial distributions for multilevel models of plant communities. pages 69–80, Calgary, Alberta, 2002. Proceedings of Graphics Interface.

[56] A Lindenmayer. Mathematical models for cellular interaction in development. In *Journal of Theoretical Biology, 18*, pages 280–315, 1968.

[57] S. Lloyd. Least squares quantization in pcm. *Information Theory, IEEE Transactions on*, 28(2):129–137, 1982.

[58] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 163–169, New York, NY, USA, 1987. ACM.

[59] N. Macdonald. *Trees and networks in biological models.* J. Wiley and sons, New York, NY, USA, 1983.

[60] Peter MacMurchy. The use of subdivision surfaces in the modeling of plants. Master's thesis, Department of Computer Science, University Of Calgary, Alberta, April 2004.

[61] B. Mandelbrot, editor. *The Fractal geometry of nature*. W.H. Freeman and Co., San Francisco, 1983.

[62] Michael McCool and Eugene Fiume. Hierarchical poisson disk sampling distributions. In *Proceedings of the conference on Graphics interface '92*, pages 94–105, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc.

[63] L. Mercer, P. Prusinkiewicz, and J. Hanan. The concept and design of a virtual laboratory. In *Proceedings on Graphics interface '90*, pages 149–155, Toronto, Ont., Canada, Canada, 1990. Canadian Information Processing Society.

[64] Don P. Mitchell. Spectrally optimal sampling for distribution ray tracing. *SIGGRAPH Comput. Graph.*, 25(4):157–164, 1991.

[65] Radomír Měch and Przemyslaw Prusinkiewicz. Visual models of plants interacting with their environment. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 397–410, New York, NY, USA, 1996. ACM.

[66] Boris Neubert, Thomas Franken, and Oliver Deussen. Approximate image-based tree-modeling using particle flows. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, page 88, New York, NY, USA, 2007. ACM.

[67] H. Nishimura, A. Hirai, T. Kawai, T. Kawata, I. Shirikawa, and K. Omura. Object modeling by distribution function and a method of image generation. *Computer Graphics Journal*, 16(2):157–160, 1985.

[68] Makoto Okabe, Shigeru Owada, and Takeo Igarashi. Interactive design of

botanical trees using freehand sketches and example-based editing. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Courses*, page 18, New York, NY, USA, 2006. ACM.

[69] Peter E. Oppenheimer. Real time design and animation of fractal plants and trees. In *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 55–64, New York, NY, USA, 1986. ACM.

[70] Kruszewski P and Whitesides S. A general random combinatorial model of botantical trees. 1998.

[71] Wojtek Palubicki, Kipp Horel, Steven Longay, Adam Runions, Brendan Lane, Radomer Mech, and Przemyslaw Prusinkiewicz. Self-Organizing tree models for image synthesis. In *SIGGRAPH '09: Proceedings of the 36th annual conference on Compute r graphics and interactive techniques*. ACM, 2009.

[72] Les Piegl and Wayne Tiller. *The Nurbs Book*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[73] Tiller W. Piegl, L. Algorithm for approximate nurbs skinning. In *Computer Aided Design*, pages 699–706, 1997.

[74] Przemyslaw Prusinkiewicz, Mark James, and Radomír Měch. Synthetic topiary. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 351–358, New York, NY, USA, 1994. ACM.

[75] Przemyslaw Prusinkiewicz, Mark James, and Radomír Měch. Synthetic topiary. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 351–358, New York, NY, USA, 1994. ACM.

[76] Przemyslaw Prusinkiewicz and Aristid Lindenmayer. *The algorithmic beauty of plants*. Springer-Verlag New York, Inc., New York, NY, USA, 1996.

[77] Przemyslaw Prusinkiewicz, Aristid Lindenmayer, and James Hanan. Development models of herbaceous plants for computer imagery purposes. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 141–150, New York, NY, USA, 1988. ACM.

[78] Przemyslaw Prusinkiewicz, Lars Mündermann, Radoslaw Karwowski, and Brendan Lane. The use of positional information in the modeling of plants. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 289–300, New York, NY, USA, 2001. ACM.

[79] S.P. Raya and J.K. Udupa. Shape-based interpolation of multidimensional objects. *Medical Imaging, IEEE Transactions on*, 9(1):32–42, Mar 1990.

[80] William T. Reeves and Ricki Blau. Approximate and probabilistic algorithms for shading and rendering structured particle systems. In *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pages 313–322, New York, NY, USA, 1985. ACM.

[81] Yodthong Rodkaew, Prabhas Chongstitvatana, Suchada Siripant, and Chidchanok Lursinsap. Particle systems for plant modeling. In *Proceedings of PMA03, Hu B.-G., Jaeger M., (Eds.)*, pages 210–217, Tsinghua University Press and Springer, Beijing, 2003.

[82] Sheldon M. Ross. *Introduction to Probability Models, Ninth Edition.* Academic Press, Inc., Orlando, FL, USA, 2006.

[83] Adam Runions, Brendan Lane, and Przemyslaw Prusinkiewicz. Modeling trees with a space colonization algorithm. In *Proceedings of the 2007 Eurographics Workshop on Natural Phenomena*, pages 63–70, 2007.

[84] Vladimir Savchenko, Er A. Pasko, Oleg G. Okunev, and Tosiyasu L. Kunii. Function representation of solids reconstructed from scattered surface points and contours. *Computer Graphics Forum*, 14:181–188, 1995.

[85] Ryan Schmidt. Interactive modeling with implicit surfaces. Master's thesis, Department of Computer Science, University Of Calgary, Alberta, August 2006.

[86] Ryan Schmidt and Brian Wyvill. Generalized sweep templates for implicit modeling. In *GRAPHITE '05: Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, pages 187–196, New York, NY, USA, 2005. ACM.

[87] Adrian Secord. Weighted voronoi stippling. In *NPAR '02: Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering*, pages 37–43, New York, NY, USA, 2002. ACM.

[88] Jonathan Shade, Michael F. Cohen, and D Mitchell. Tiling layered depth images. Technical report, University of Washington, Department of Computer Science, 2000.

[89] K. Shinozaki, K. Yoda, and K. Hozumi. A quantitative analysis of plant form the pipe model theory. i. basic analyses. In *Japanese Journal of Ecology 14*, pages 97–104, 1964.

[90] Jos Stam. Aperiodic texture mapping. Technical report, European Research Consortium for Informatics and Mathematics (ERCIM, 1997.

[91] A. Takenaka. A simulation model of tree architecture development based on growth response to local light environment. In *Journal of Plant Research*, pages 321–330, 1994.

[92] Greg Turk. Generating random points in triangles. *Graphics gems*, pages 24–28, 1990.

[93] Greg Turk and James F. O'Brien. Shape transformation using variational implicit functions. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Courses*, page 13, New York, NY, USA, 2005. ACM.

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

[95] Robert Ulichney. *Digital halftoning*. MIT Press, Cambridge, MA, USA, 1987.

[96] Hao Wang. Proving theorems by pattern recognition i. *Commun. ACM*, 3(4):220–234, 1960.

[97] J. Weber and J. Penn. Creation and rendering of realistic trees. In *SIGGRAPH '95: Conference Proceedings*, pages 119–128. ACM, 1996.

[98] Turner Whitted. An improved illumination model for shaded display. *Commun. ACM*, 23(6):343–349, 1980.

[99] F. Wither, J. amd 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.

[100] C. Woodward. Methods for cross-sectionals design of B-spline surfaces. In *EuroGraphics*, pages 129–142, 1986.

[101] C. Woodward. Skinning techniques for interactive B-spline surface interpolation. In *Computer-Aided design*, pages 441–451, 1988.

[102] Brian Wyvill, Eric Galin, and Andrew Guy. Extending The CSG Tree. Warping, Blending and Boolean Operations in an Implicit Surface Modeling System. *Computer Graphics Forum*, 18(2):149–158, June 1999.

[103] Geoff Wyvill, Craig McPheeters, and Brian Wyvill. Data Structure for Soft Objects. *The Visual Computer*, 2(4):227–234, February 1986.

[104] John I. Yellott, Jr. Spectral consequences of photoreceptor sampling in the rhesus retina. *Science*, 221:382 – 385, July 1983.

[105] Gary Yngve and Greg Turk. Robust creation of implicit surfaces from polygonal meshes. *IEEE Transactions on Visualization and Computer Graphics*, 8(4):346–359, 2002.