UNIVERSITY OF CALGARY

A Biomechanical Model of Branch Shape in Plants Expressed Using L-Systems

by

Catherine Alena Jirasek

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

CALGARY, ALBERTA

JULY, 2000

Abstract

Environmental factors play an essential role in the development of plants. Specifically, the initial shape of a plant axis may be modified by mechanical and physiological factors. From a mechanical perspective, the bending of an axis results from external forces that are applied to it, such as the force of gravity. From a physiological perspective, curvature change of an axis may be initiated by a tropic influence [18]. In this thesis, a biomechanical model of axis shape that is affected by mechanical and physiological processes is developed. The model is suitable for realistic synthesis of plant images in computer graphics applications, and is implemented using the L-system methodology [41] which provides a theoretical framework and software environment for developing plant architectures. The model of axis shape is extended from individual axes to branching structures, and is extended to include collisions between plant elements and between an axis and mechanical obstacles.

Acknowledgments

The enclosed thesis could not have been completed without the inspiration of several people. First and foremost, I would like to thank my supervisor, Dr. Prusinkiewicz, for his patience, motivation, and enthusiasm throughout the stages of our research. Dr. Prusinkiewicz taught me well and guided me to completion of this thesis. I would like to thank my family for their unconditional support throughout my academic career. Thank you to my parents for the uplifting encouragement when things were looking down; thank you to my brother for the thought provoking conversations that would uncover new paths to be explored. I would like to thank my fiance, Giuseppe Montagnese, who taught me how to balance all aspects of life while keeping my priorities in sight. And last but not least, I would like to thank all of those who had confidence in my abilities and who provided me with encouragement and motivation.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Statement of the Problem

Environmental factors play an essential role in the development of plants. Specifically, the initial orientation and shape of a plant axis may be modified by mechanical and physiological factors. From a mechanical perspective, the bending of an axis results from the external forces that are applied to it, such as the force of gravity, the forces due to collisions between plant axes, and the collisions between an axis and a mechanical obstacle. The magnitude of the response of an axis to the external forces depends on its rigidity. For example, terminal branches of a weeping willow are very elastic and hang down due to gravity. Larger branches are more rigid and have a smaller tendency to bend.

These purely mechanical phenomena are combined with the physiological phenomena of curvature change due to the differential growth of tissues on the opposite sides of an axis [46]. Differential growth may be initiated by a tropic influence that is an environmental stimulus with a strong directional component that regulates the growth of an organ [18]. For example, the growth pattern of plant organs is often affected by light; a stem may turn toward a light source, whereas a leaf may assume an orientation that is perpendicular to the direction of incoming light.

A plant axis grows in the longitudinal and radial dimensions. Longitudinal growth occurs when the axis segments, or internodes, increase in length or are added at the distal end of an axis. Radial growth is due to the addition of layers of material to the circumference of an axis. It produces an increase in girth, an increase in weight, and an increase in rigidity of the axis. As we discuss in detail in Chapter 5, radial growth also results in branch shape memory; if all external forces were removed from a developed axis, the axis would partially maintain the bending and overall shape that it had achieved over the natural growth process.

The key objective of our work was to develop a model of plant axis shape suitable for realistic synthesis of plant images in computer graphics applications. To this end, we adapted a biomechanical model proposed by Fournier *et al.* [14], which integrates mechanical and physiological phenomena outlined above. Rod theory is used to capture purely mechanical effects of the external forces on an axis. Longitudinal growth, radial growth, and tropisms represent the related physiological factors. We extend Fournier's results by considering the development of entire branches, as opposed to individual axes, and by considering collisions that may occur between plant axes, or between an axis and a mechanical obstacle.

Another contribution of this work was the implementation of the proposed biomechanical mechanism using the formalism of L-systems [41]. L-systems provide a good theoretical framework and software environment for modeling the architecture of plants. They have been used to capture a variety of interactions between plants and their environment [29], but the effects of gravity and tropisms have not been represented in previous L-system models in a satisfactory manner. Our implementation overcomes this limitation. The results are illustrated by models and realistic images of several plants and plant structures.

1.2 Overview of Sections

The thesis is organized as follows. Background and previous work are discussed in Chapter 2. A continuous model of an elastic rod that is placed in a gravitational field is analyzed in Chapter 3. The discretized version of the continuous model is presented in Chapter 4. The mechanical model that is presented in Chapter 4 is extended to a biomechanical model in Chapter 5, where longitudinal growth of an axis, radial growth of an axis, and tropisms are discussed. In Chapter 6, the model of axis growth and bending is implemented using the L-system methodology. In Chapter 7, two cases of collisions are considered: collisions between elements of a plant axis and collisions between an axis and a mechanical obstacle. The implementation of the collision models is discussed as well. Resulting images are presented in Chapter 8, and concluding remarks are made in Chapter 9.

Chapter 2

Background

2.1 Terminology

2.1.1 Axial Trees

An axial tree consists of branch segments that begin at the tree base and form paths up to the terminal nodes [41] (see Figure 2-1). A segment that is followed by at least one more segment is referred to as an internode, whereas a segment with no succeeding edges is referred to as an apex. At a node, we distinguish at most one straight segment; the remaining segments are referred to as lateral segments. A plant axis consists of a sequence of straight internodes, where the first internode originates at the root or as a lateral segment at a node. An axis may support higher-order lateral axes, which, together, constitute a branch.

2.1.2 Order in Trees

We adopt the ordering method for axes that was proposed by Gravelius [26]. The method begins at the root of a tree and traces a path towards the terminal nodes. An axis originating at the root has an order of zero, whereas an axis that branches from an n-order parent axis has an order of $n + 1$ [41] (see Figure 2-1).

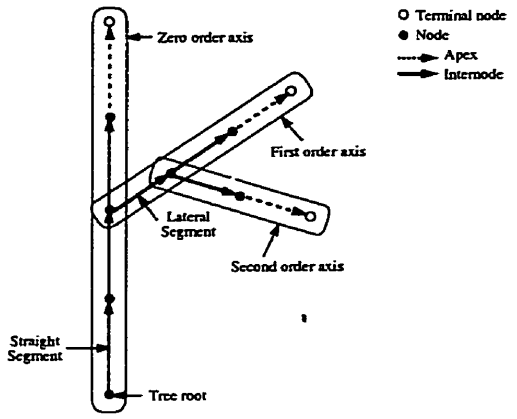


Figure 2-1: An axial tree and the adopted ordering scheme. Redrawn from [41].

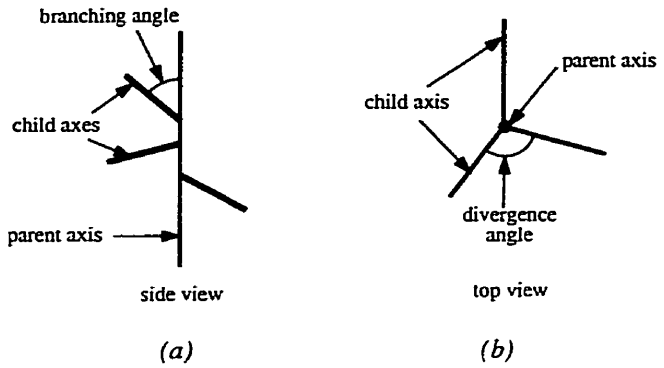


Figure 2-2: (a) A child axis forms a branching angle with respect to its parent branch. (b) A child axis forms a divergence angle with neighboring child axes.

2.1.3 Initial Orientation of an Axis

A child axis forms a branching angle with respect to its parent branch, and forms a divergence angle with neighboring child axes [41] (see Figure 2-2). The initial values of these angles depend on the plant species and the location of branches. For example, axes may diverge at an angle of 137.5° about the parent branch, which is referred to as a spiral phyllotactic pattern; a distichuous pattern refers to axes that are placed at 180° with respect to each other, and a decussate pattern refers to axes that are rotated by 90° with respect to each other [49] (see Figure 2-3).

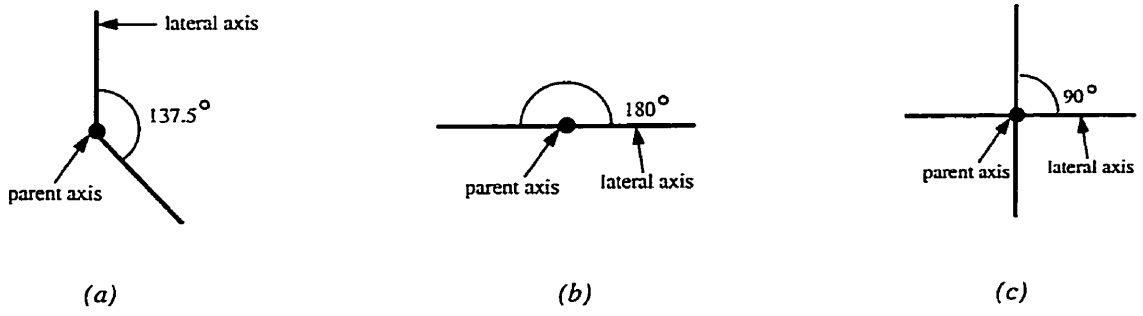


Figure 2-3: Top views of: (a) a spiral phyllotactic pattern, (b) a distichuous pattern, (c) a decussate pattern.

2.1.4 Reorientation and Movement of an Axis

Over time, the initial orientation of an axis may be modified by mechanical or physiological factors. The reorientation of plant axes is usually gradual, but the resulting change in shape of branches and positions of supported axes can be significant.

In the purely mechanical case an axis bends due to external forces. For example, the twigs of a weeping birch or a weeping willow are flexible and will hang down due to the force of gravity.

An axis may also bend as a result of a physiological change in the structure of the axis. Specifically, a curvature change due to differential growth occurs when one side of the axis grows faster or slower than the other side [46]. When the opposite sides have unequal lengths, the axis curves towards the shorter side.

Differential growth of an axis may result from internal or external causes. If the differential growth of an axis is determined by an internal process inherent in the plant, the movement is termed nastic [18]. If the curving is determined by an external stimulus that regulates axis growth, the movement is termed tropic [48]. The present work considers the movement and reorientation of plant axes caused by eternal stimuli.

2.1.5 Tropisms

Tropic movements result in axes being positioned such that they are best adapted to the conditions of their environment. In a tropic response, the differential growth within an axis is initiated and regulated by an environmental stimulus that, generally, has a strong directional component [18]; the change in orientation or movement of an axis is related to the direction of the environmental stimulus.

In a tropic response, different axes may assume quite different positions with respect to the external stimulus [48]. For example, a branch or stem may grow towards a light source, whereas leaves may assume orientations perpendicular to the direction of light. The direction of a tropic response of an axis is described using various terms [18]. If an axis turns in the direction parallel to the stimulus, the response is described as orthotropic. If an axis bends to assume a right angle to the stimulus then the response is diatropic, whereas if an axis bends to assume any other angle to the direction of the stimulus then the response is plagiotropic. A response is said to be positive if it seeks the source of the stimulus, and negative if it moves away from the stimulus [18].

Tropisms are named according to the environmental factors that initiate them [18]. The two major tropic responses in plants are gravitropism and phototropism. Gravitropic responses are those in which a plant axis assumes a definite position with respect to the direction of gravity [48]. A vertical line is used as a standard reference orientation, and the value of the angle between the axis and the vertical line characterizes different types of gravitropic responses [18]. Phototropic responses are those in which a plant axis assumes a definite position with respect to a light source, where heliotropism specifically refers to sunlight as the orienting stimulus.

Other less common tropisms also affect the orientation of plant axes. For example, thigmotropism is a directional response to physical contact or touch, traumatropism is a response to axis injury, chemotropism is a response to a chemical stimulus, and hydrotropism is a response to large quantities of moisture [18].

2.2 Related Work

2.2.1 Axis Orientation Due to Gravity and Tropisms

Many biological studies have been conducted to improve our understanding of the effects of gravity and tropisms on the growth of plant axes.

Suzuki and Kitano conducted a study that showed that shoot growth is affected by the orientation of its parent stem with respect to gravity [50]. Initially, Suzuki and Kitano identified that two phases of growth activity occur: the sprouting phase in which the growth of shoots did not depend on the orientation of the parent stem and the dominance phase in which the growth of shoots did depend on the orientation of the stem. In the dominance phase, the growth of lower shoots was suppressed in vertically oriented stems. In arched stems, the distal shoots grew less vigorously and the upper lateral shoots began to assert dominance over the growth of the lower lateral shoots (see Figure 2-4). Suzuki and Kitano suggested that the suppression of the growth of shoots is affected by its orientation with respect to gravity.

Several studies have focused on the gravitropic response of different plant species. Bjorkman identified that gravitropism is the result of four different steps [4]. The first step in

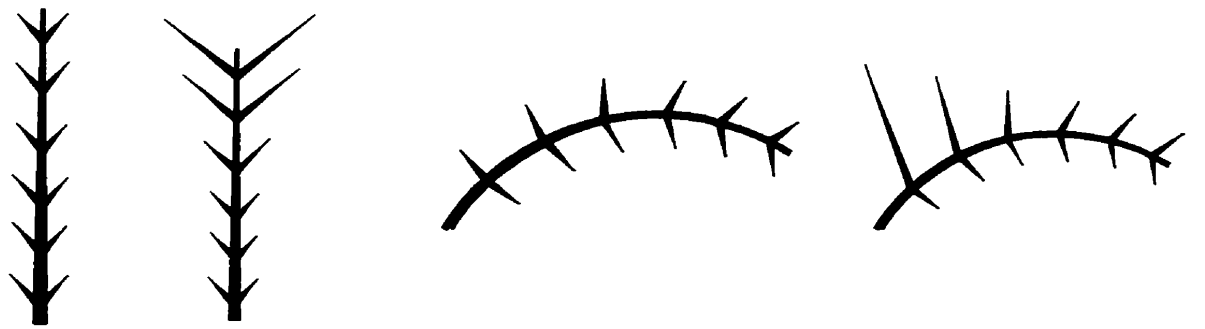


Figure 2-4: (a) Erect stem during the sprouting phase. (b) Erect stem during the dominance phase. (c) Arched stem during the sprouting phase. (d) Arched stem during the dominance phase. Redrawn from [50].

gravitropism, called susception, is the initial physical reaction resulting from gravity of an element in a plant. Due to the gravitational force, an element in a plant, referred to as a sensing particle, will move which causes a physiological response in the plant. The conversion of the physical signal to a physiological signal is called perception and is the second step in gravitropism. Gravity sensing occurs in certain regions of a plant, but the curvature change due to gravity often occurs in different regions where growth is controlled. The third step in gravitropism, called transmission, involves the communication of a signal from the cells where gravity is sensed to the cells where growth occurs. The final step in gravitropism, called the response, is the resulting differential growth on opposite sides of a plant axis. The curvature change resulting from differential growth due to a gravitropic response was analyzed by Kohji *et al.* [23]. The bending that resulted from gravitropism was caused by an extension growth of the convex side of an axis and the contraction of the concave side of the axis.

Bennet-Clark and Ball showed that certain plant species are genuinely diagravitropic (they have a tendency to grow in a horizontal direction) [2]. Bennet-Clark and Ball performed several experiments where they initially displaced axes at different upward and downward orientations. They observed the growth pattern following the artificial displacement and found that rapid upward and downward movement occurred after the initial transposition of the axes. The oscillations gradually stabilized and the axes continued to grow horizontally. Bennet-Clark *et al.* also performed similar experiments on roots that initially grew vertically [3]; after an artificial change in orientation the roots underwent a rapid curvature change which eventually subsided, and the roots resumed growth in a vertical direction.

Kohji *et al.* showed that the gravitropic response of an axis can reverse throughout different developmental stages [23]. They found that the gravitropic response of a peduncle (the main stalk of flowers) in certain plant species changes from negative to positive after the flowering phase. This shift in the gravitropic response illustrates the ability of plant bodies to modify their response to environmental stimuli according to the stages of development. A related positive-to-negative change in the gravitropic response of an axis was analyzed by Digby and Firn, who referred to this phenomenon as a gravitropic sign reversal [31].

Throughout a gravitropic sign reversal response, an internode passes through tropic phases that result in angles in-between the initial and final orthogravitropic orientations [10]. By displacing an internode above or below its current position, Digby and Firn found that the internode would show a dramatic gravitropic response by returning to the developmental angle that it achieved before reorientation. Digby and Firn concluded that each internode can sense gravity throughout its development, and an internode has the ability to restore its orientation to a developmentally defined angle. Therefore, the majority of plant axes maintain an orientation that is not vertical and can pass through phases that are conventionally referred to as negative orthogravitropism, negative plagiogravitropic, diagravitropic, positive plagiogravitropic, and positive orthogravitropic (see Figure 2-5a). Subsequently, Digby and Firn substituted these descriptive terms with a numerical gravitropic set-point angle (GSA), which was defined as the angle with respect to gravity that is maintained by an axis as a consequence of gravitropism (see Figure 2-5b). The GSA concept provides a unifying idea that suggests that different orientations resulting from gravitropic responses need not be considered as being governed by different mechanisms [11].

Using the results of these biological studies, several models have been developed that describe and simulate the effects of gravity and tropisms on the orientation of axes.

Prusinkiewicz *et al.* introduced a method for incorporating tropic tendencies into branching models [42]. The bending of branches that are affected by tropisms can be simulated by slightly rotating each branch segment in the direction of the tropic influence. The angle

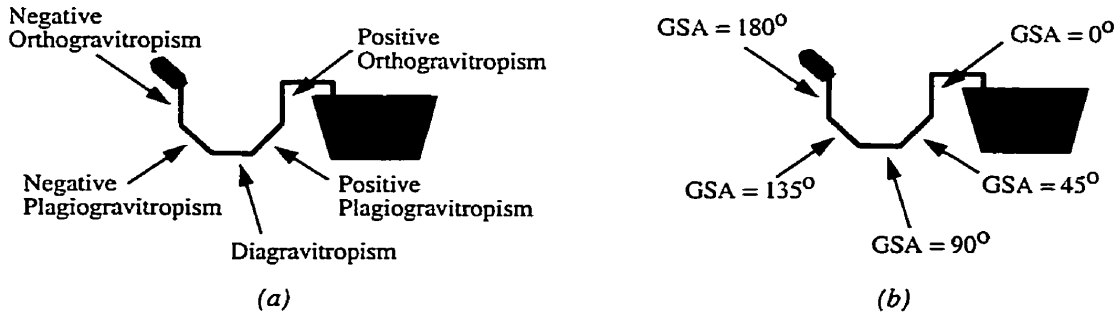


Figure 2-5: (a) Conventional terms used to describe certain forms of gravitropism, (b) The equivalent GSA values. Redrawn from [10].

of orientation α is calculated using the formula $\alpha = e|\vec{H} \times \vec{T}|$ where e is a proportionality coefficient that captures the magnitude of the segments tropic response, $\vec{H}$ is the orientation of a branch segment, and $\vec{T}$ is the direction of the tropism (see Figure 2-6). The formula is motivated by a physical analogy: $\vec{T}$ is thought of as a force applied to the end of vector $\vec{H}$, thus $\vec{H} \times \vec{T}$ represents the torque applied to segment $\vec{H}$ that rotates about its starting point. A similar approach for incorporating tropic tendencies into branching models was used later by Holton [20].

Chiba *et al.* extended this method to simulate the reorientation of branches due to their local light environment [5] [6]. A branch is placed at an orientation similar to the direction of a phototropic vector, which is parallel to the brightest direction of light within the atmosphere. To find the brightest direction with respect to a branch, the amount of sunlight that reaches the branch is approximated. Each branch has an associated cluster of leafs, referred to as a leaf-ball. The amount of sunlight that reaches a leaf-ball is estimated by initially determining the area of the shadows projected from each leaf-ball onto a celestial sphere that encloses the branching structure. The amount of sunlight that reaches a leaf-ball depends on the amount of overlap that occurs between a leaf-ball and the surrounding leaf-balls. The brightest direction vector is the sum of the vectors from the celestial sphere that reach the leaf-ball. Each branch segment is slightly rotated toward the brightest direction vector, as was done by Prusinkiewicz *et al.* [42].

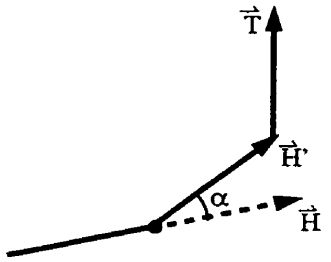


Figure 2-6: Due to a tropism $\vec{T}$, segment $\vec{H}$ is oriented around its starting point by angle α. Redrawn from [41].

2.2.2 Gravitropism and Branching

The following discussion reviews models that incorporate the effects of gravitropism on branching structures. Although gravitropism is related to branch orientation in the following reviews, the results have not been utilized further in our research, which focused on the orientation of an individual axis as opposed to entire branches.

Honda *et al.* developed and expanded three basic models of branching, where two of the models include the effects of gravitropism. In all three models, referred to as the horizontal plane model (H-model), the perpendicular plane model (P-model), and the inclined plane model (I-model), a parent branch bifurcates into two child branches. In the H-model, branches bifurcate such that the child branches and the parent branch create a plane. The branch plane is chosen such that it is closest to the horizontal plane, that is, it has the least possible slope (see Figure 2-7a) [22].

The direction of gravity does not affect the branching patterns in the basic P-model [22]. The branch plane of the initial parent branch P_0 and the child branches P_1 and P_2 is chosen in the same manner as in the H-model. Each successive branch plane is placed at a perpendicular orientation to its parent branch plane. That is, the plane that P_1, and child

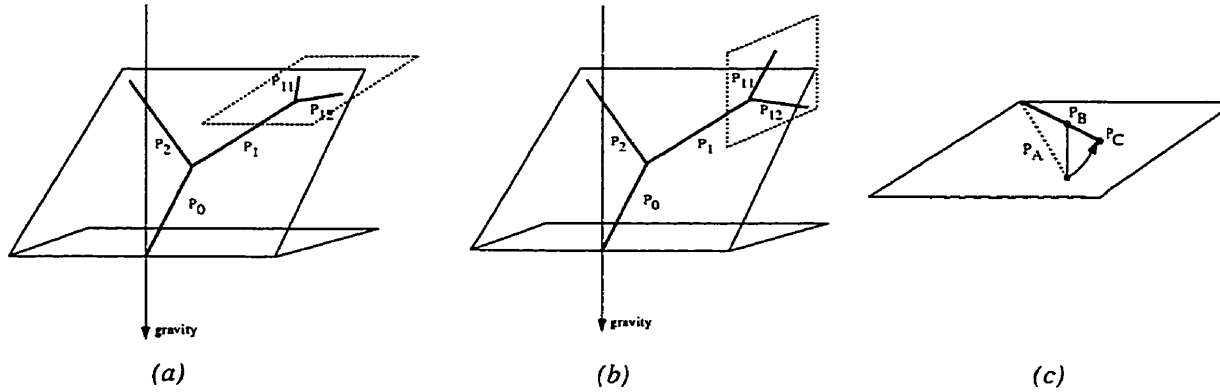
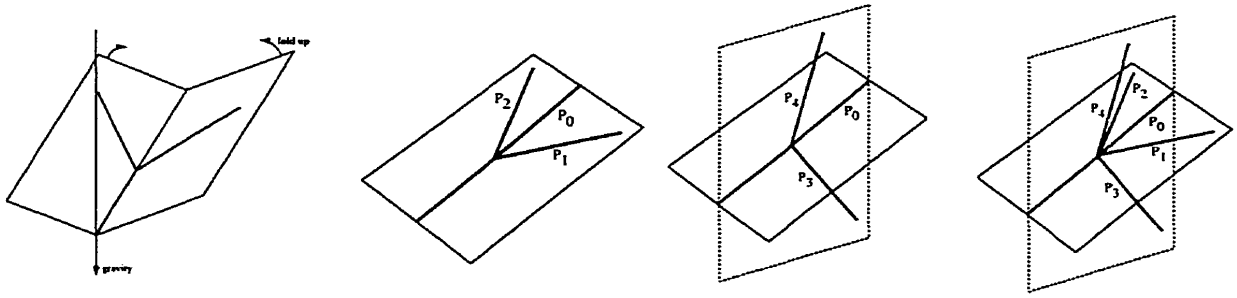


Figure 2-7: (a) In the H-model, a branching plane is oriented such that it is closest to the horizontal plane. (b) In the P-model, each successive branching plane is placed at a perpendicular orientation to its parent branch plane. (c) The P-model is modified to include gravitropism by projecting a branch that is initially oriented downward onto the horizontal plane. The new branch is directed along the projected branch, but maintains the length of the original branch. Redrawn from [22].

branches P_{11} and P_{12} reside in is perpendicular to the plane that P_0, P_1, and P_2 reside in (see Figure 2-7b). Honda *et al.* expanded the basic P-model to take into account negative gravitropic effects on each branch. When a branch is oriented below the horizontal plane, it is rotated upward such that it lies in the horizontal plane. That is, a downward oriented branch P_A is projected onto the horizontal plane to produce a temporary branch P_B. The final resulting branch P_C that is affected by a negative gravitropism has the direction of P_B and the length of the original branch P_A (see Figure 2-7c).

The I-model that was introduced by Honda *et al.* also incorporates negative gravitropism on child branches [21]. Initially, the parent and child branches lie in a plane that is closest to the horizontal plane (as defined in the H-model). Gravitropism is included in the model by folding the plane upward along the direction of the parent branch (see Figure 2-8a). Honda *et al.* expanded the I-model to allow a parent branch to produce five child branches that are affected by negative gravitropism. Each child branch lies in either a horizontally oriented plane (as defined in the H-model) or a vertical plane, where the two planes are perpendicular to each other. Child branch P_0 is an extension of the parent branch. Child branches P_1 and P_2 lie in the horizontally oriented plane, and child branches P_3 and P_4 lie in the vertical plane (see Figure 2-8b). Branches P_1 and P_2 are affected by gravitropism by folding the horizontally oriented plane upward along the direction of the parent



(a) (b)

Figure 2-8: (a) Gravitropism is included in the I-model by folding the horizontal branching plane upward along the direction of the parent branch. (b) The I-model was expanded to allow a parent branch to produce five child branches that are affected by negative gravitropism. Each child branch lies in either a horizontally oriented plane or a vertical plane, where the two planes are perpendicular to each other. Redrawn from [21].

branch. The central branch P_0 and branches P_3 and P_4 are rotated more vertically against gravity.

2.2.3 Radial Growth of a Plant Axis

Shinozaki *et al.* introduced a model, commonly referred to as the pipe model, that quantitatively characterizes the girth of axes [45]. They postulated to consider an axis to be made up of a bundle of pipes of constant thickness or cross-sectional area. Each pipe runs from a leaf, through all of the successive axes, down to the trunk base. The cross-sectional area of an axis internode is proportional to the number of pipes that run through the internode (see Figure 2-9). Thus, the sum of the cross-sectional areas of stems and branches that are found at a certain horizontal level is proportional to the number of leaves that have existed at and above that level.

The pipe model proposes a mechanistic explanation to observations that were made dating back to Leonardo daVinci [47]. da Vinci postulated that "all branches of trees at every stage of their height, united together, are equal to the thickness of their trunk". Several other researchers have tested the applicability of the pipe model theory. For example, Shinozaki *et al.* [44] and Chiba [7] both examined the forms of trees and found that their results supported the concept of the pipe model theory.

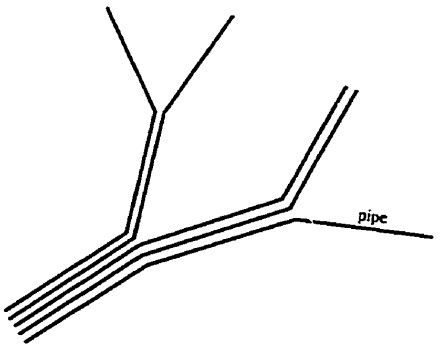


Figure 2-9: The pipe model. Redrawn from [20].

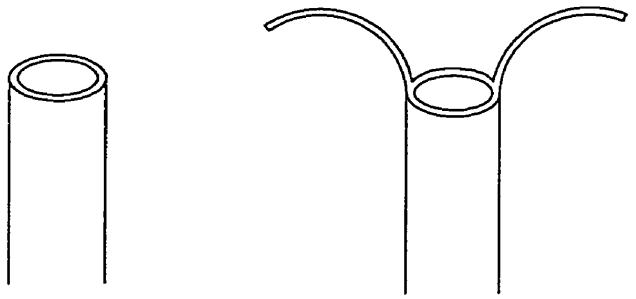


Figure 2-10: (a) An intact vertically oriented plant axis. (b) The bending of sectors due to a non-zero torque exerted by the forces that generate the tissue stresses in the sector. Redrawn from [19].

2.2.4 Biomechanical Studies and Models Related to Axis Orientation

Studies have been conducted that focused on the biomechanics behind axis movement and reorientation. Hejnowicz studied tissue stresses (internal forces applied to a tissue), both tensile and compressive, that act on a tissue layer in a plant axis [19]. Tissue stresses may be a result from either differential growth of tissues or from the structural variations between tissues within a plant axis. The bending and orientation of a plant axis is a result of the magnitude and direction of the tissue stresses within an axis. In a vertically oriented axis, the forces which generate tissue stresses exert large bending torques that sum to zero. Since the torques sum to zero, the vertical plant axis is in a state of equilibrium which results in no bending or repositioning of the axis. Hejnowicz showed this phenomenon by slitting a stem into several longitudinal sectors. He observed that each sector would bend due to the non-zero torque which was being exerted by the tissue stresses. However, an intact vertical stem did not bend because the total torque was zero (see Figure 2-10). Hejnowicz also observed the mechanical change of a horizontally oriented axis. During a positive gravitropic response of an axis, the tensile tissue stresses decreased on the lower side of the axis while those on the upper side remained unchanged. The change in tissue stresses resulted in a curvature change of the axis.

Niklas examined how the properties of different plant materials influence the mechanical behavior of plants [33]. He compared the attributes of plant materials to those of fabricated materials and found that many plant materials behave as if they were elastic solids. Niklas used the theory of elasticity to determine the spatial distribution of forces within structures, which in turn affects the shape and movement of a plant axis.

Fournier *et al.* used the theory of elasticity to analyze the effects of biological and environmental factors on the mechanical properties of a plant [14]. Their analysis considered the change in mechanical state of each point along a stem during an infinitesimal time period. In particular, Fournier *et al.* considered the kinetics of a gravitationally strained form of a radially and longitudinally growing stem (in two-dimensions) that was affected by an upward tropism. Young axis internodes were affected by a tropic influence. As the stem grew older, layers of material were consecutively added to the internodes, resulting in an increase in rigidity of the stem. Over time, a given part of the stem would tend to lean downward due to its own weight. The resulting shape of the stem was an "S" due to the combined effects of age, load, and tropisms. Details of Fournier's analysis of axis shape are elaborated on throughout the thesis.

Specific tree architectures that incorporated Fournier's model were produced by Fourcaud with the AMAP software [13]. Faurcaud did not have the flexibility of defining arbitrary plant structures, therefore, the models that he produced were specific to certain plant architectures. The implementation of the biomechanical model of axis shape that is presented in the thesis does not have this restriction.

Chapter 3

Theory of a Continuous Rod

3.1 Stresses and Strains

A plant can change the properties of its materials in response to mechanical forces that are caused by growth or the external environment. Due to this change over time, the mechanical properties of most plant materials are difficult to quantify [33]. We approximate the behavior of plant materials as if they were ideal elastic rods, with the assumption that the rod as a whole can not stretch. Elastic solid rods are used as the initial approximation in our model of a plant axis.

Any combination of forces that are applied to a rod can be decomposed into two fundamental force components: the normal force and the tangential force [33]. The normal force, which acts perpendicular to a cross-section of the rod, results in either tension or compression of the rod. The tangential force, which acts parallel to a cross-section, results in shear deformations of the rod. A surface that runs from end to end of a deformed rod that is neither contracted nor extended is referred to as the neutral surface [1].

When an external force is applied to a rod, internal forces develop throughout the rod [33]. Stress is defined as the force per area over which the internal forces act [8]:

$$\delta = \frac{dF}{dA}, \tag{3-1}$$

where δ is the stress, dF is the force, and dA is the area.

Stresses that operate perpendicular to each cross-section of a rod are referred to as normal stresses, whereas stresses that operate parallel to the plane of each cross-section are referred to as shear stresses.

Strain refers to the intensity of the deformation of a rod that results from stress components [33]. The normal strain, resulting from normal stresses, reflects the ratio of the change in deformation to that of the undeformed state:

$$\xi = \frac{l - l_0}{l_0}, \tag{3-2}$$

where ξ is the normal strain, l_0 is the original dimension and l is the deformed dimension. Shear strain, resulting from shear stresses, reflects the change in the original right angle between two axes at a point.

3.2 The Force and Torque Acting on an Infinitesimal Segment Caused by External Forces

To obtain a description of the shape of a rod in equilibrium that is subject to external forces, initially consider an infinitesimal rod segment of length dl which is bound by two cross-sections A and B (see Figure 3-1a). Let $-\vec{F}$ be the stress force that is acting on cross-section A, $\vec{F} + d\vec{F}$ be the stress force that is acting on cross-section B, and $\vec{S}$ be the external force per unit length that is acting on the rod. When the rod is in a state of equilibrium, the sum of these forces must equal zero

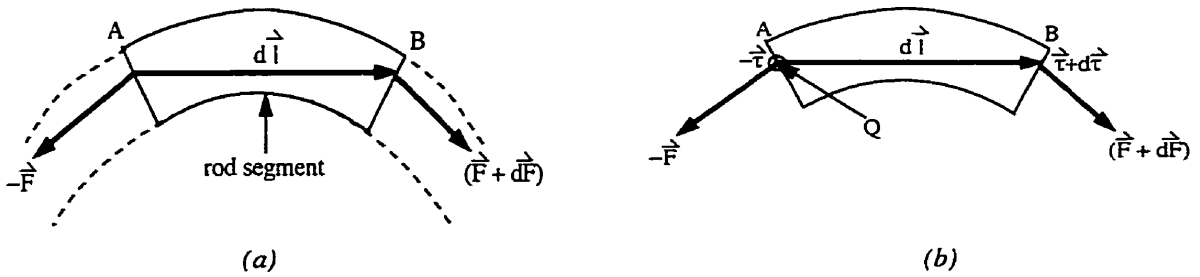


Figure 3-1: (a) An infinitesimal segment of length dl with stress forces acting on each cross-section. (b) The total torque about an arbitrary point Q depends on both the force and the torque due to the over hanging parts of the rod.

$$-\vec{F} + \vec{F} + d\vec{F} + \vec{S}dl = 0 \ , \tag{3-1}$$

thus

$$\frac{d\vec{F}}{dl} = -\vec{S} \ . \tag{3-2}$$

The equation for the torque of an infinitesimal rod segment about an arbitrary point Q depends on the external force $\vec{S}$ acting on the segment, and the stress force and torque acting on the segment due to the over-hanging parts of the rod. Let Q be placed at the midpoint of cross-section A. Let $-\vec{F}$ and $\vec{F} + d\vec{F}$ be the stress forces that are acting on cross-sections A and B, respectively, let $-\vec{\tau}$ be the torque of cross-section A about Q, and let $\vec{\tau} + d\vec{\tau}$ be the torque of cross-section B about Q (see Figure 3-1b). The equilibrium equation for the total torque about Q is

$$-\vec{\tau} + \vec{\tau} + d\vec{\tau} + \vec{0} \times -\vec{F} + d\vec{l} \times (\vec{F} + d\vec{F}) + d\vec{l} \times \vec{S}dl = 0 \ , \tag{3-3}$$

$$d\vec{\tau} + d\vec{l} \times \vec{F} + d\vec{l} \times d\vec{F} + d\vec{l} \times \vec{S}dl = 0 \ . \tag{3-4}$$

Ignoring higher-order infinitesimals, the equation becomes:

$$d\vec{t} + d\vec{l} \times \vec{F} = 0 \ . \tag{3-5}$$

By dividing by the length dl, and by using the fact that $d\vec{l}/dl = \hat{r}$ is the unit tangent vector to the rod at a given point, the final equation becomes

$$\frac{d\vec{t}}{dl} = -(\hat{r} \times \vec{F}) \ . \tag{3-6}$$

Throughout the remaining chapters of this thesis, we parameterize the rod from right to left (as opposed to the left to right convention used above) which results in a change in sign of the right hand side of equations (3-2) and (3-6). Therefore, the equations that are used from here on in are:

$$\frac{d\vec{F}}{dl} = \vec{S} \ , \tag{3-7}$$

$$\frac{d\vec{t}}{dl} = \hat{r} \times \vec{F} \ . \tag{3-8}$$

3.3 Rigidity of an Elastic Rod

3.3.1 Flexural Rigidity of a Rod

The flexural rigidity specifies the ability of an object to resist bending and is defined to be the product of the elastic modulus and the moment of area [33]. To derive this relationship in two-dimensions, consider the bending of a rod segment that has a neutral surface of length dl. The extension of a surface with length dl' that is a distance x from the neutral surface is related to the radius of curvature by the following ratio (see Figure 3-2)

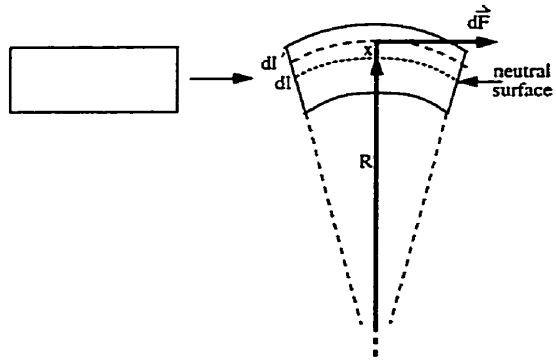


Figure 3-2: A force $d\vec{F}$ that is acting on a layer with length dl' that is a distance x from the neutral surface.

$$\frac{dl'}{dl} = \frac{R+x}{R} = 1 + \frac{x}{R} . \tag{3-9}$$

The normal strain ξ is obtained by manipulating equation (3-9) to get:

$$\frac{dl'-dl}{dl} = \xi = \frac{x}{R} . \tag{3-10}$$

Generalized Hooke's Law [8] describes the relationship between the stress δ and strain ξ:

$$\delta = E\xi , \tag{3-11}$$

where E is the elastic modulus of the object. Substituting equation (3-10) into equation (3-11) gives

$$\delta = E\frac{x}{R} . \tag{3-12}$$

Recall that the normal force dF acting on an infinitesimal area in the rod is

$$dF = \delta dA , \tag{3-13}$$

where dA is the area over which the stress force acts. By substituting equation (3-12) into

equation (3-13), we obtain:

$$dF = \left(E\frac{x}{R}\right)dA \ . \tag{3-14}$$

The force dF that is acting on an infinitesimal area that is a distance x from the neutral surface produces a torque $d\tau$ (see Figure 3-2):

$$d\tau = xdF \ . \tag{3-15}$$

Substituting equation (3-14) into (3-15) gives

$$d\tau = \frac{E}{R}x^2dA \ . \tag{3-16}$$

Integrating this equation over the entire cross-section A produces

$$\tau = \frac{E}{R}\int_A x^2dA \ , \tag{3-17}$$

$$= \frac{EI}{R} \tag{3-18}$$

$$= EIK \ , \tag{3-19}$$

where I is the moment of area and $K = \frac{1}{R}$. In the two-dimensional case, $K = \frac{d\theta}{dl}$ is the curvature of the rod, where $d\theta$ is the infinitesimal change in orientation of a segment with length dl. In the three-dimensional case, the change in orientation can be represented as a rotation vector since the change is infinitesimal [27]. The rotation vector can be decomposed along the principal axes [9] of the rod segment, which is done in Section 3.4.

3.3.2 Torsional Rigidity of a Rod

The treatment of torsional rigidity, which is the resistance to torsion of the rod, is analogous to the treatment of flexural rigidity [33]. The torsional rigidity C is defined to be the product of the shear modulus G and the torsional constant J:

$$C = GJ \ . \tag{3-20}$$

The torsional constant is a mathematical expression of the distribution of the material in a cross-section. It is defined to be the amount of torque required to produce a rotation of one radian per unit length of a beam, divided by the shear modulus [33].

3.4 The Equations of Equilibrium of a Rod

The final torque acting on an infinitesimal segment in a rod depends on two quantities: the torque due to external forces $d\vec{\tau}_{external}$ and the torque due to the elasticity of the rod $d\vec{\tau}_{elastic}$. At static equilibrium, the equation

$$d\vec{\tau}_{external} + d\vec{\tau}_{elastic} = \vec{0} \tag{3-21}$$

must hold for every infinitesimal segment along the rod. Since the flexural and torsional rigidity values differ about each principal axis of an infinitesimal segment P, we decompose equation (3-21) along the unit x, y, and z axes at P. The decomposition of $d\vec{\tau}_{external}$ about the x, y, and z axes, respectively, is

$$d\vec{\tau}_{x_{external}} = (dl\vec{r} \times \vec{F}) \cdot \hat{x} \ , \tag{3-22}$$

$$d\vec{t}_{y_{external}} = (dl\vec{r} \times \vec{F}) \cdot \hat{y} \ , \tag{3-23}$$

$$d\vec{t}_{z_{external}} = (dl\vec{r} \times \vec{F}) \cdot \hat{z} \ , \tag{3-24}$$

and the decomposition of $d\vec{t}_{elastic}$ about the x, y, and z axes, respectively, is

$$d\vec{t}_{x_{elastic}} = C_x \frac{d\theta_x}{dl} \ , \tag{3-25}$$

$$d\vec{t}_{y_{elastic}} = C_y \frac{d\theta_y}{dl} \ , \tag{3-26}$$

$$d\vec{t}_{z_{elastic}} = C_z \frac{d\theta_z}{dl} \ , \tag{3-27}$$

where $\vec{F}$ is the stress force acting on the segment, $\hat{r}$ is the unit tangent vector at P, dl is the length of the infinitesimal segment, C_x is the torsional rigidity of the segment about the x-axis, C_y and C_z are the flexural rigidities about the y and z axes, respectively, and $\frac{d\vec{\theta}}{dl} = \left(\frac{d\theta_x}{dl}, \frac{d\theta_y}{dl}, \frac{d\theta_z}{dl}\right)$ is the infinitesimal change in orientation of a segment with length dl. Equation (3-21) coupled with equations (3-22) to (3-27) are the equations that must be solved to find the static equilibrium of a rod.

Chapter 4

A Discrete Model of Plant Axis Shape

4.1 A Discretization of the Continuous Rod Model

4.1.1 Basic Concept of the Model

The continuous model that was presented in the previous chapter can be discretized to calculate the final shape of a rod subject to external forces. Two approaches of the discretization are pursued: a discretization is derived from the continuous formulas presented in Chapter 3, and an elementary derivation is presented for a discrete linkage system connected by rotational springs. The two methods are produced for completeness of presentation.

A plant axis is assumed to consist of discrete segments connected by rotational springs. Each segment is assigned a length, width, mass, and local coordinate system. The mass of a segment is assumed to be concentrated at the end of the segment. Segments are connected together by three springs at a joint. Each spring has one degree of freedom about a local coordinate axis. Together, the three springs behave as one individual spring that has three degrees of rotational freedom. Three individual spring constants will specify the flexibility of each spring about an axis. The plant axis as a whole is subject to gravity and, possibly, other predefined external forces.

4.1.2 The Force Acting on a Segment

4.1.2.1 A Derivation of the Force Acting on a Segment from the Continuous Rod Model

The numerical methods for finding the static equilibrium of an axis subject to external forces involves the discretization of the continuous rod model. By considering a straight-forward discretization of the continuous model that was presented in the previous chapter, equation (3-7) becomes:

$$\Delta \vec{F} = \vec{S} \Delta l \ , \tag{4-1}$$

where $\Delta \vec{F}$ is the stress force acting on the segment and $\vec{S}$ is the external force per unit length that is acting on a rod segment of length Δl.

4.1.2.2 An Elementary Derivation of the Force Acting on a Segment

The force acting on a segment of length Δl can be derived from the equation

$$\Delta m = \rho A \Delta l \ , \tag{4-2}$$

where Δm is the mass of the segment, ρ is the density of the segment, A is the cross-sectional area of the segment, and Δl is the length of the segment. Both A and ρ are assumed to be constant over the entire segment. By multiplying both sides of equation (4-2) by an acceleration $\vec{a}$, the equation becomes:

$$\Delta \vec{F} = \rho A \vec{a} \Delta l = \vec{S} \Delta l \ , \tag{4-3}$$

where $\vec{S}$ is the external force per unit length.

4.1.3 The Torque Acting About a Joint Caused by External Forces

4.1.3.1 A Derivation of the Discrete Torque Formula from the Continuous Rod Model

By considering a straightforward discretization of the continuous model that was presented in the previous chapter, equation (3-8) becomes:

$$\Delta\vec{\tau}_{external} = \Delta l\vec{r} \times \vec{F} \ , \tag{4-4}$$

where $\vec{F}$ is the stress force due to the overhanging part of the rod.

We parametrize the discretized model such that joint i is at the beginning of the axis, joint 0 is at the end of the axis, and the intermediate joints have indices that are decreasing consecutively from the beginning to the end of the axis (see Figure 4-1). Let $\vec{\tau}_i$ denote the torque acting about a joint i, let $\vec{r}_{i-1}$ denote the vector connecting joint i and joint $i-1$, and let $\vec{F}_i$ denote the force acting on segment i. These three vectors are specified in a global coordinate system. Equation (4-4) can then be written in the form

$$\vec{\tau}_{i_{external}} = \vec{r}_{i-1} \times \sum_{j=0}^{i-1} \vec{F}_j + \vec{\tau}_{i-1_{external}} \tag{4-5}$$

for the torque about joint i of an axis consisting of i segments.

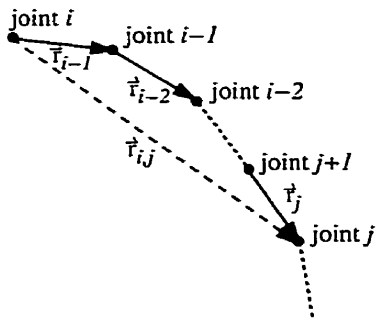


Figure 4-1: The graphical interpretation of a rod consisting of several discrete segments.

4.1.3.2 An Elementary Derivation of the Discrete Torque Formula

The discrete torque formula can also be obtained directly by considering an axis as a linkage of stiff elements connected by rotational springs. For completeness of presentation, an elementary derivation of the discrete torque formula is presented below. This second method of derivation is found in areas of robotics [9], whereas the first method of derivation is found in elasticity theory.

The total torque about joint i of an axis consisting of i segments is the sum of the torques due to forces that are applied to the end of each segment in the axis:

$$\vec{t}_{i_{external}} = \sum_{j=0}^{i-1} \vec{r}_{i,j} \times \vec{F}_j \,, \tag{4-6}$$

$$= \vec{r}_{i,i-1} \times \vec{F}_{i-1} + \sum_{j=0}^{i-2} \vec{r}_{i,j} \times \vec{F}_j \,. \tag{4-7}$$

where $\vec{r}_{i,j}$ denotes the vector connecting joint i and joint j (see Figure 4-1). By definition,

$$\vec{r}_{i,j} = \vec{r}_{i,i-1} + \vec{r}_{i-1,j} = \vec{r}_{i-1} + \vec{r}_{i-1,j} \,. \tag{4-8}$$

By substituting equation (4-8) into equation (4-7) we obtain

$$\vec{t}_{i_{external}} = \vec{r}_{i-1} \times \vec{F}_{i-1} + \sum_{j=0}^{i-2} (\vec{r}_{i-1} + \vec{r}_{i-1,j}) \times \vec{F}_j \tag{4-9}$$

$$= \vec{r}_{i-1} \times \vec{F}_{i-1} + \sum_{j=0}^{i-2} (\vec{r}_{i-1} \times \vec{F}_j + \vec{r}_{i-1,j} \times \vec{F}_j) \tag{4-10}$$

$$= \vec{r}_{i-1} \times \vec{F}_{i-1} + \sum_{j=0}^{i-2} (\vec{r}_{i-1} \times \vec{F}_j) + \sum_{j=0}^{i-2} (\vec{r}_{i-1,j} \times \vec{F}_j) \ . \tag{4-11}$$

By recognizing that the torque about joint $i-1$ is

$$\vec{\tau}_{i-1_{external}} = \sum_{j=0}^{i-2} \vec{r}_{i-1,j} \times \vec{F}_j \ , \tag{4-12}$$

we can relate recursively $\vec{\tau}_i$ to $\vec{\tau}_{i-1}$:

$$\vec{\tau}_{i_{external}} = \vec{r}_{i-1} \times \vec{F}_{i-1} + \sum_{j=0}^{i-2} (\vec{r}_{i-1} \times \vec{F}_j) + \vec{\tau}_{i-1_{external}} \tag{4-13}$$

$$= \vec{r}_{i-1} \times \vec{F}_{i-1} + \vec{r}_{i-1} \times \sum_{j=0}^{i-2} \vec{F}_j + \vec{\tau}_{i-1_{external}} \tag{4-14}$$

$$= \vec{r}_{i-1} \times \sum_{j=0}^{i-1} \vec{F}_j + \vec{\tau}_{i-1_{external}} \ , \tag{4-15}$$

which is the same as equation (4-5).

4.1.4 Finding the Projection of the Torque Along Each Axis of a Segments Local Coordinate System

Each segment has a local coordinate system associated with it. The coordinate system is oriented along the principal axes of the cross-section of the segment so that the torque vector that was found in the previous section can be projected onto each local axis. This gives the magnitude of the torque about each axis which, in turn, is used to find the final rotational displacement of the segment about each axis.

The x-axis of the coordinate system for segment $i-1$ is parallel to segment i (see Figure 4-2). The y and z axes are chosen as the principal axes that produce a right-handed coordinate system. The projection of the torque $\vec{\tau}_i$ onto the local x-axis $\vec{e}_{i_x}$ is given by

$$\vec{\tau}_{i_x} = \left(\frac{\vec{\tau}_i \cdot \vec{e}_{i_x}}{\|\vec{e}_{i_x}\|^2} \right) \vec{e}_{i_x} \ . \tag{4-16}$$

The projection of the torque onto the local y and z axes can be found in a similar manner.

4.1.5 Introducing the Torque of the Spring

To keep a spring displaced from its rest state, a force must be applied to each end of the spring. As the force increases in magnitude, the elongation of the spring will increase. If the elongation is not too great, the magnitude F of the force that is being applied to the spring is directly proportional to the displacement x of the spring from its rest state:

$$F = -kx \ , \tag{4-17}$$

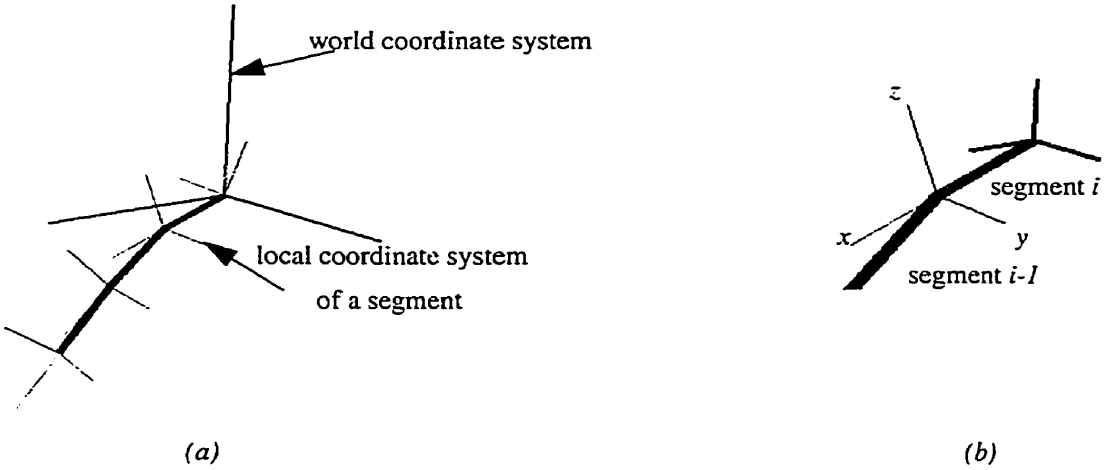


Figure 4-2: (a) Each segment is rotated about a local coordinate system, where the local x-axis for segment $i-1$ assumes an orientation that is parallel to the previous segment. (b) A close-up view of the local coordinate system.

where k is the spring constant that is a measure of the material properties of the spring [33]. Equation (4–17) is known as Hooke's Law [51].

In the continuous model, a torque was exerted due to the flexural and torsional rigidity of the rod. Similarly, in the discrete case, the spring at joint i that connects segments i and $i-1$ exerts a torque in the negative direction to the torque $\hat{\boldsymbol{\tau}}_{i_{external}}$ in equation (4-15). The magnitude of the torque exerted by the spring at joint i can be found using the rotational analogue of Hooke's law. In the three-dimensional case, the spring will have three degrees of rotational freedom about each local axis. Since, in general, the rigidity of the spring about each axis differs, there will be an individual torque value due to the spring about each axis. For example, the torque due to the spring about the local x-axis at joint i is

$$\tau_{i_{x-spring}} = -\kappa_{i_x}\theta_{i_x}\,, \tag{4-18}$$

where κ_{i_x} is the constant that specifies the rigidity of the spring about the local x-axis and θ_{i_x} is the angle about the local x-axis between the springs rest state and the springs current state. The torque exerted by the spring about the local y and z axes can be found in a similar manner.

4.1.6 A Relationship Between the Length and Spring Constant of a Segment

The rigidity of the spring at joint i depends on the length of segment $i-1$. This relationship is derived as follows.

The torque τ due to a spring is

$$\tau = \kappa\Delta\theta\,, \tag{4-19}$$

where κ is the spring constant and $\Delta\theta$ is the displacement of the spring from its rest state. From equation (3-19), it is known that

$$\tau = EIK = EI\frac{\Delta\theta}{\Delta l}, \tag{4-20}$$

where E is the elasticity modulus, I is the moment of area, and $K = \frac{\Delta\theta}{\Delta l}$ is the change in orientation of a segment with length Δl.

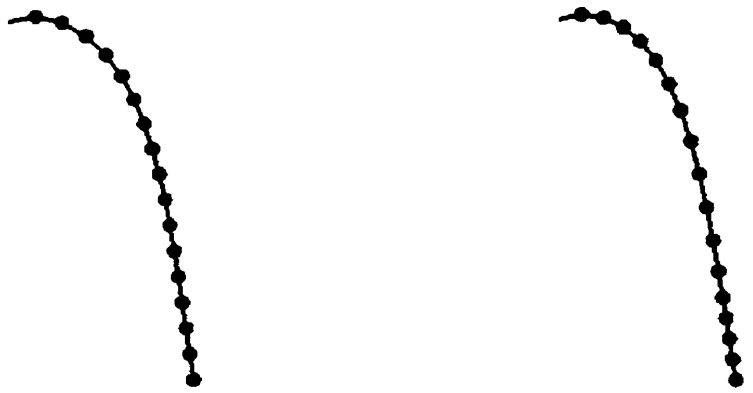
From a comparison of equation (4-20) and equation (4-19), it follows that the spring constant at joint i that approximates the elastic properties of a segment $i-1$ with length l_{i-1} is equal to

$$\kappa_i = \frac{E_i I_{i-1}}{l_{i-1}}. \tag{4-21}$$

Thus, the spring constant about each local axis of a joint i is inversely proportional to the length of segment $i-1$:

$$\kappa_i \sim \frac{1}{l_{i-1}}. \tag{4-22}$$

Figure 4-3a illustrates an example of an axis that consists of equal length segments, whereas Figure 4-3b illustrates an example of an axis that has shorter end segments and longer intermediate segments. The curves in Figure 4-3 are approximately the same shape, which confirms that both models are valid discretizations of the continuous rod.



(a) (b)

Figure 4-3: (a) An example of an axis that consists of equal length segments. (b) An example of an axis that consists of longer intermediate segments.

4.2 Numerical Methods for Finding the Equilibrium of an Axis

4.2.1 Considering the Frictional Damping Torque in a Dynamic System

A dynamical relaxation method [34] is used to obtain a final state of static equilibrium of an axis. In a dynamic system, a damping torque caused by the medium in which the axis bends acts in the opposite direction to the angular velocity of a segment [51]. The damping torque can be broken down into three components about each local axis. For example, the damping torque about the local x-axis at joint i is given by:

$$\tau_{i_{x-damping}} = -b\omega_{i_x}, \tag{4-23}$$

where b is the constant that specifies the strength of the damping torque and ω_{i_x} is the angular velocity about the local x-axis of segment $i-1$ with respect to segment i. The damping torque about the y and z axes can be found in a similar manner.

Taking into account the spring torque $\tau_{i_{spring}}$, the damping torque $\tau_{i_{damping}}$, and the torque

due to external forces $\tau_{i_{external}}$, the total torque about each axis at joint i is:

$$\tau_{i_{x-total}} = \tau_{i_{x-external}} + \tau_{i_{x-spring}} + \tau_{i_{x-damping}}, \tag{4-24}$$

$$\tau_{i_{y-total}} = \tau_{i_{y-external}} + \tau_{i_{y-spring}} + \tau_{i_{y-damping}}, \tag{4-25}$$

$$\tau_{i_{z-total}} = \tau_{i_{z-external}} + \tau_{i_{z-spring}} + \tau_{i_{z-damping}}. \tag{4-26}$$

4.2.2 Finding the Final Rotational Angle of a Segment

Using dynamics, the angle that segment $i-1$ makes with respect to segment i can be found as the angle at which the axis is in static equilibrium. Consider the dynamic system of segment i and segment $i-1$. The frame of reference attached to segment i is noninertial since the segment may be accelerating or rotating about an axis due to a force that is being applied to it. If the motion of segment $i-1$ is considered with respect to that frame of reference, then pseudo forces will be acting on segment $i-1$. For simplicity, these pseudo forces are ignored since they do not affect the equilibrium state of the segments (in equilibrium, there is no motion and no acceleration, which, in turn, means that there are no pseudo forces acting on the segments).

Three systems of differential equations are solved numerically to find the final rotational displacement of a segment about each local axis. To derive the system of differential equations, initially consider the rotation of segment $i-1$ about its local x-axis. Knowing the torque about joint i, the angular acceleration of segment $i-1$ can be found using the equation [51]

$$\alpha_{i_x} = \frac{\tau_{i_{x-total}}}{I_{i_x}}, \tag{4-27}$$

where I_{i_x} is the x component of the moment of inertia of the axis about joint i. Although the axis is not a rigid body, we approximate its moment of inertia as if it were a rigid body; this approximation will not affect the validity of the static solution that will be obtained. Thus, the x component of the moment of inertia about joint i is calculated as

$$I_{i_x} = \sum_{j=0}^{i-1} m_j R_{j_x}^2 , \tag{4-28}$$

where m_j is the mass attached to the end of segment j and R_{j_x} is the distance from the end of segment j to the x-axis of rotation (see Figure 4-4). The sum is taken from $j = 0$ to $i-1$.

The angular acceleration of segment $i-1$ is related to its position θ_{i_x} (the angle of rotation about the local x-axis of segment $i-1$ with respect to segment i) and angular velocity ω_{i_x} by equations [51]:

$$\frac{d\theta_{i_x}}{dt} = \omega_{i_x} , \tag{4-29}$$

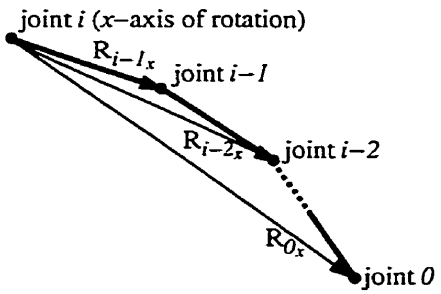


Figure 4-4: The moment of inertia about the x-axis is calculated using the distance R_{j_x} from the end of segment j to the center of rotation.

$$\frac{d\omega_{i_x}}{dt} = \alpha_{i_x} , \tag{4-30}$$

where t is time. By expressing the acceleration α_{i_x} using equation (4-27), and substituting equation (4-24) for $\tau_{i_{x-total}}$, we obtain a system of differential equations from which we can find the equilibrium state of the axis as a solution to the initial value problem:

$$\frac{d\theta_{i_x}}{dt} = \omega_{i_x} , \tag{4-31}$$

$$\frac{d\omega_{i_x}}{dt} = \frac{\tau_{i_{x-external}} + \tau_{i_{x-spring}} + \tau_{i_{x-damping}}}{I_{i_x}} . \tag{4-32}$$

We find this solution numerically using (for simplicity) Euler's method [36]:

$$\theta_{j_x}^{k+1} = \theta_{j_x}^{k} + \omega_{j_x}^{k} \Delta t , \tag{4-33}$$

$$\omega_{j_x}^{k+1} = \omega_{j_x}^{k} + \alpha_{j_x}^{k} \Delta t \qquad \text{for } j = 1, 2, \ldots, i , \tag{4-34}$$

where k is the number of numerical iterations. The initial angle $\theta_{i_x}^{0}$ of segment $i-1$ about joint i is set to a predefined elevation angle. It is assumed that the initial values $\theta_{j_x}^{0}$ for $j = 1, 2, \ldots, i-1$ are set to 0 with respect to $\theta_{i_x}^{0}$, and the initial velocity $\omega_{j_x}^{0}$ for $j = 1, 2, \ldots, i$ are set to 0. The displacement of a segment about its local y and z axes can be found in a similar manner.

Figure 4-5 shows an example of an axis that is subject to the force of gravity.

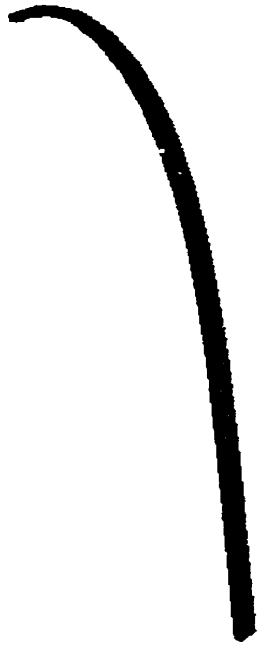


Figure 4-5: An example of an axis that is subject to the force of gravity.

Chapter 5

An Extension of the Mechanical Model of Axis Bending to a Biomechanical Model of Axis Growth

5.1 The Simulation of the Growth of an Axis

The mechanical model of axis bending that was presented in the previous chapter can be used to simulate the growth of a plant axis.

An axis will be subject to forces and will assume a predefined length for each segment and predefined spring constants for each joint. The simulation of the growth of a plant axis begins with one segment. The local coordinate system of the segment is set to the initial orientation of the segment, with the x-axis parallel to the segment. For k iterations, Euler's method is used to find the change in position of the segment about the local x, y, and z axes. The segment is rotated about each axis by the calculated angles. When the equilibrium state has been obtained, a new segment that initially assumes the same orientation as the previous segment is added to the end of the axis. The local coordinate system of the new segment is set to coincide with the orientation of the previous segment. That is, the local x-axis is parallel to the previous segment, and the local y and z axes are set as principal axes such that a right-handed coordinate system is obtained. The change in posi-

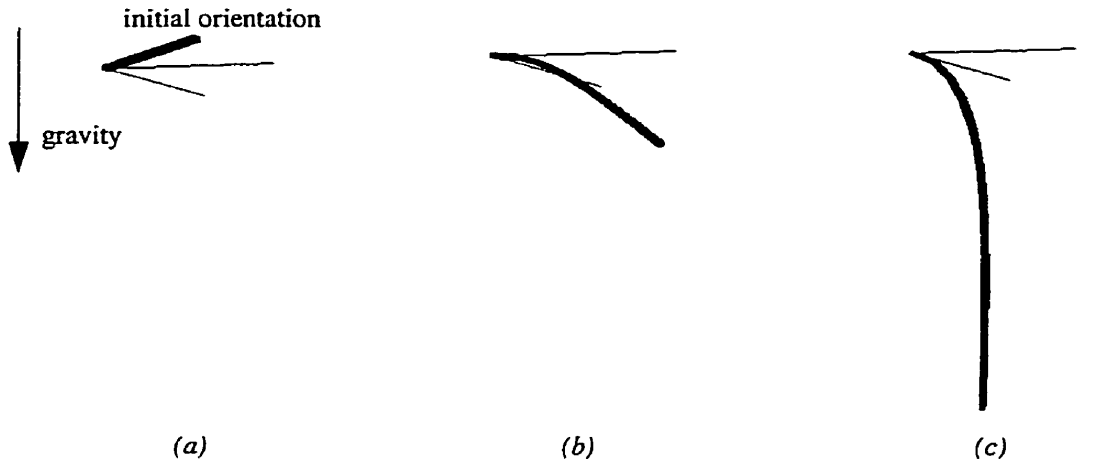


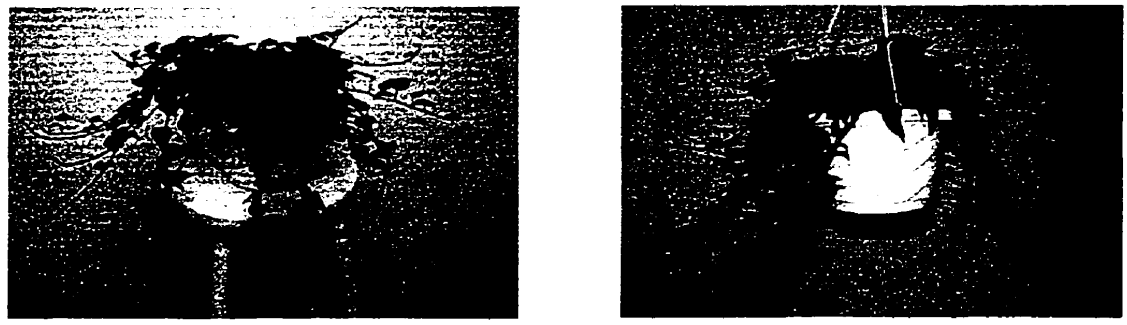
Figure 5-1: The growth of an axis that is subject to the force of gravity.

tion of each segment is calculated until a state of equilibrium is reached, and the process repeats for any number of segments. Figure 5-1 shows an example of the growth of an axis that is subject to the force of gravity.

5.2 Incorporating Tropisms into the Model

As discussed by Digby and Firn [10], the organs of many plant species have a tendency to grow in a specific direction. The effect of tropisms on a plant axis was used in the context of modeling by Fournier *et al.* [14]. We adopt Fournier's method and simulate the effect of tropisms by rotating new segments in the direction of the preferred tendency of growth. Over time, the orientations of the internodes are altered due to external forces acting on the axis, such as the force of gravity. Figure 5-2 shows an example of plant axes that have a tendency to grow in an upward direction but that are subject to the force of gravity.

The differential growth of a plant axis that is a result of tropisms can be approximated as the growth of an axis that is subject to a force oriented in the direction of the preferred tendency of growth. In our model, tropisms are represented as a force vector that affects the orientation of newly added segments. In the case of a gravitropic influence, the tropic vec-

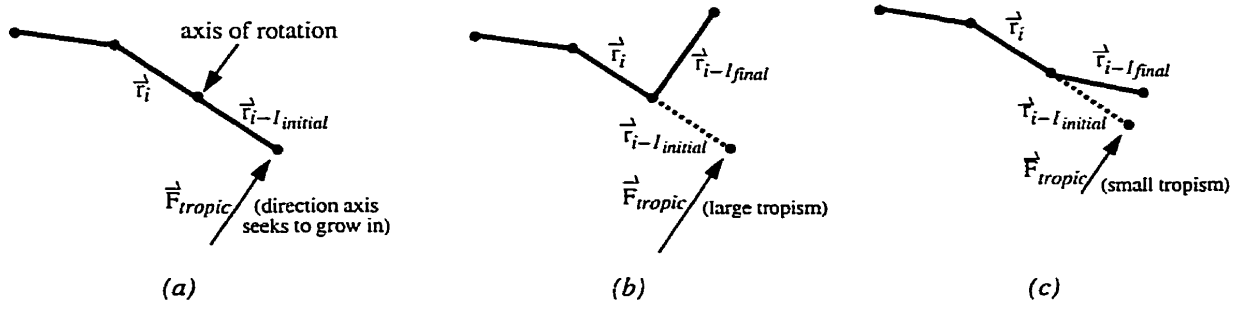


(a) (b)

Figure 5-2: Plant axes that seek to grow in an upward direction but that are subject to the force of gravity.

tor is oriented vertically. In the case of a phototropic influence, the tropic vector may have an arbitrary orientation. In both instances, the tropic vector influences all axes that are in a branching structure; the case where axes may be in shadow of other axes is not considered in the phototropic case.

In our model, the orientation of a newly added segment is affected by the tropism that is acting on the axis. Initially, assume that a new segment $i-1$, represented by vector $\vec{r}_{i-1}$, is attached at an orientation that is parallel to the previous segment i. A tropism $\vec{F}_{tropic}$ is applied to the end of the segment which causes the segment to revolve about an axis of rotation (see Figure 5-3). A very large tropism will displace the segment such that it is parallel to the direction that the axis seeks to grow in; a small tropism will orient the segment such that it is closer to the desired orientation, but may not entirely reach the desired orientation. The orientation that a new segment assumes is referred to as the rest state or the rest angle of the segment.



(a) (b) (c)

Figure 5-3: (a) A tropism is applied to the end of a newly added segment; the segment is able to rotate about its axis of rotation. (b) A very large tropism will orient the segment such that it is parallel to the tropism vector. (c) A small tropism will orient the segment such that it is closer to the desired orientation, but the segment may not entirely reach the desired orientation.

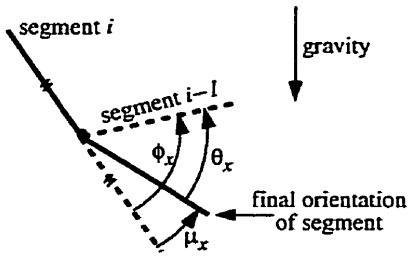


Figure 5-4: Due to gravity, segment $i-1$ is rotated by angle θ_x with respect to its rest angle ϕ_x. The resulting angle μ_x between segment i and $i-1$ is equal to $\phi_x - \theta_x$.

The torque $\vec{\tau}_{i_{tropic}}$ acting about joint i that is a result of $\vec{F}_{tropic}$ acting on segment $i-1$ is

$$\vec{\tau}_{i_{tropic}} = \vec{r}_{i-1} \times \vec{F}_{tropic} \,. \tag{5-1}$$

Given the torque due to the tropism, the rotational displacement of the segment about each local axis can be found in the same manner as was discussed in Section 4.2.2; $\vec{\tau}_{i_{tropic}}$ is projected onto each local axis and three systems of differential equations are solved numerically to find the rest angle of the segment about each axis.

The rest state is calculated when a new segment is attached to the end of an axis. During the simulation, the final orientation of a segment depends on both its rest state as well as its displacement due to external forces that are acting on the axis. For example, a segment is rotated about its local x-axis by $\phi_x - \theta_x$ where θ_x is the angular displacement due to external forces and ϕ_x is the rest angle of the segment (see Figure 5-4). Assuming that angles are small, the orientation about the y and z axes can be found in a similar manner. Figure 5-5 illustrates the growth of an axis that is subject to gravity as well as a phototropism.

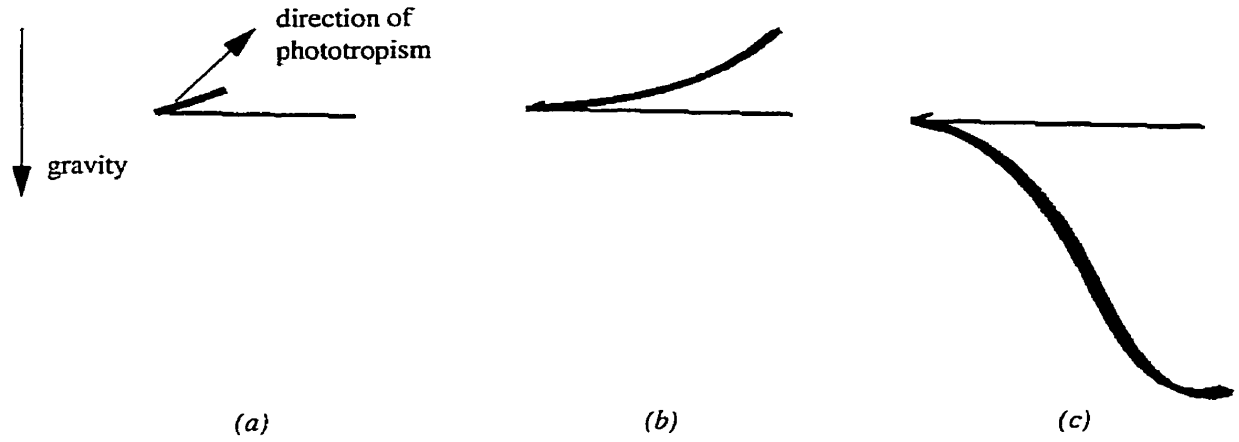


Figure 5-5: The growth of an axis that is subject to gravity as well as a phototropism.

5.3 Radial Growth of a Segment

Not only do axes grow longitudinally, they also grow radially. We capture the radial growth using the pipe model introduced by Shinozaki *et al.* [45]. Each segment is thought of as a collection of concentric pipes, where all pipes have the same cross-sectional area. When a new segment is attached to the end of an axis, each previous segment will grow in a radial direction by obtaining an additional bundle of pipes. The newly added pipes form an outer layer of material which results in secondary growth of each segment [43] (see Figure 5-6). The same cross-sectional area is added to each segment when a new layer is added to the axis.

The addition of layers to an axis will affect three properties: the rest angle of each segment will be altered, the material properties of the axis will change, and the mass of each segment will increase.

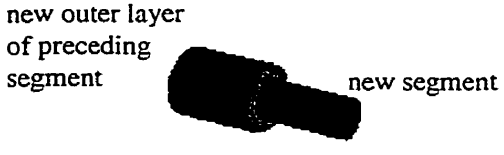


Figure 5-6: The longitudinal and radial growth of an axis. When a new segment is attached to the end of an axis, previous segments will grow and form a new outer layer of material.

Following the model introduced by Fournier *et al.* [14], the rest state of each segment in a radially growing axis will be altered due to the addition of new layers of material to the segment. Each new layer (or bundle of pipes) will have a distinct rest angle about each local axis, which, in turn, will affect the rest angle of the segment as a whole. When new layers are added to an axis, the new segment $i-1$ that is attached to the end of the axis assumes a rest orientation resulting from the tropism. As more segments are added to the axis, segment $i-1$ will obtain new layers. Each new layer that is added to segment $i-1$ will have a rest angle about each axis equal to the current angle between segment i and segment $i-1$ (see Figure 5-7).

The radial growth of segment $i-1$ will also result in the increase of the rigidity of the spring at joint i. That is, the joint will become less elastic.

Begin with the rotational analogue of Hooke's Law to derive a relationship for the over all rest angle and spring constant about the local x-axis of a segment that consists of several layers of material. The torque exerted by an individual spring is

$$\tau_{i_x} = \kappa_{i_x}\theta_{i_x} \tag{5-2}$$

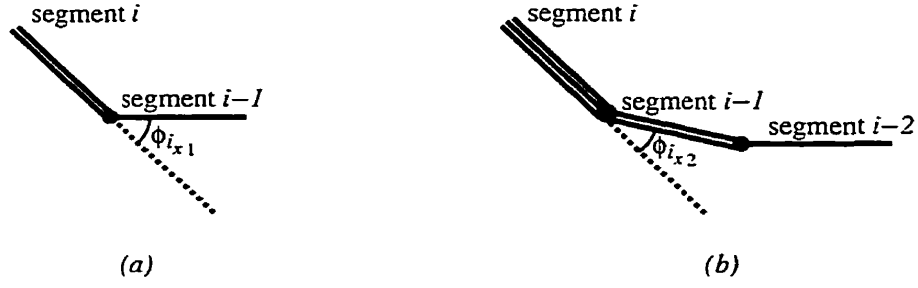


Figure 5-7: (a) Rest angle $\phi_{i_{x1}}$ of the first layer in segment $i-1$ resulting from the tropism. (b) When a segment is added to the end of an axis, the rest angle $\phi_{i_{x2}}$ of the second layer in segment $i-1$ will be equal to the current angle between segments i and $i-1$ about the local x-axis (resulting from the total torque applied to segment $i-1$).

$$= \kappa_{i_x}(\phi_{i_x} - \mu_{i_x}) \ , \tag{5-3}$$

where τ_{i_x} is the torque of spring i about the local x-axis, κ_{i_x} is the spring constant, θ_{i_x} is the rotational displacement of segment $i-1$ due to external forces, ϕ_{i_x} is the rest angle of segment $i-1$, and μ_{i_x} is the angle between segments i and $i-1$ as shown in Figure 5-4.

A collection of layers that make up a segment can be considered as a single layer. In other words, there exists a spring constant $\bar{\kappa}_{i_x}$ and a rest angle $\bar{\phi}_{i_x}$ such that the multi-layer system will behave as a single angular spring described by the equations

$$\tau_{i_x} = \bar{\kappa}_{i_x}\bar{\theta}_{i_x} \tag{5-4}$$

$$= \bar{\kappa}_{i_x}(\bar{\phi}_{i_x} - \mu_{i_x}) \ . \tag{5-5}$$

To show this, assume that a segment i consists of n layers. Since torques are additive, the resultant total torque τ_{i_x} due to each layer will be equal to:

$$\tau_{i_x} = \kappa_{i_{x1}}(\phi_{i_{x1}} - \mu_{i_x}) + \kappa_{i_{x2}}(\phi_{i_{x2}} - \mu_{i_x}) + \cdots + \kappa_{i_{nx}}(\phi_{i_{xn}} - \mu_{i_x}) \tag{5-6}$$

$$= \kappa_{i_{x1}}\phi_{i_{x1}} + \kappa_{i_{x2}}\phi_{i_{x2}} + \ldots + \kappa_{i_{xn}}\phi_{i_{xn}} - \mu_{i_x}(\kappa_{i_{x1}} + \kappa_{i_{x2}} + \cdots + \kappa_{i_{xn}}) \tag{5-7}$$

$$= (\kappa_{i_{x1}} + \kappa_{i_{x2}} + \cdots + \kappa_{i_{xn}})\left(\frac{\kappa_{i_{x1}}\phi_{i_{x1}} + \kappa_{i_{x2}}\phi_{i_{x2}} + \cdots + \kappa_{i_{xn}}\phi_{i_{xn}}}{\kappa_{i_{x1}} + \kappa_{i_{x2}} + \cdots + \kappa_{i_{xn}}} - \mu_{i_x}\right) . \tag{5-8}$$

Comparing formulas (5-8) and (5-5), we obtain:

$$\bar{\kappa}_{i_x} = \kappa_{i_{x1}} + \kappa_{i_{x2}} + \cdots + \kappa_{i_{xn}} \ , \tag{5-9}$$

$$\bar{\phi}_{i_x} = \frac{\kappa_{i_{x1}}\phi_{i_{x1}} + \kappa_{i_{x2}}\phi_{i_{x2}} + \cdots + \kappa_{i_{xn}}\phi_{i_{xn}}}{\kappa_{i_{x1}} + \kappa_{i_{x2}} + \cdots + \kappa_{i_{xn}}} . \tag{5-10}$$

Assuming that angles are small, the overall rest angle and spring constant about the local y and z axes can be calculated in a similar manner. In the simulation, each time a new layer is added to a segment the overall spring constant and rest angles are recalculated for each segment. The change in rest state affects the overall shape of the axis if all external forces were to be removed. That is, an axis that has been growing over a period of time, subject to external forces, will maintain its bending and overall shape when external forces are removed.

As a segment grows radially, the mass of the segment will increase as well. The increase in mass of a segment will be proportional to the increase in cross-section of the segment, assuming constant density throughout the axis.

5.3.1 Calculating the Spring Constant of a New Layer

As stated above, each new layer that is added to a segment has a unique spring constant that is used in the calculation of the overall spring constant at that joint. The spring constant about each local axis for an individual layer can be calculated. For example, the spring constant about the x-axis at joint i for an individual layer m is calculated using equation (4-21) which states that

$$\kappa_{i_{xm}} = \frac{E_{i_x} I_{i-1_x}}{l_{i-1}} , \tag{5-11}$$

where E_{i_x} is the shear modulus of the material about the x-axis, l_{i-1} is the length of segment $i-1$ that rotates about joint i, and I_{i-1_x} is the torsional constant of the layer, which

is calculated according to one of two cases. If the segment has just been added then the cross-section is assumed to be a full circle. The torsional constant about joint i is calculated using the formula [33]

$$I_{i-1_x} = \frac{\pi}{2} R_{i-1}^4 \ , \tag{5-12}$$

where R_{i-1} is the radius of the segment. If the segment has grown radially, then the cross-section of the new layer is assumed to be a hollow ring, and the torsional constant of the new layer is calculated using the more general formula

$$I_{i-1_x} = \frac{\pi}{2}(R_{i-1}^4 - r_{i-1}^4) \ , \tag{5-13}$$

where R_{i-1} is the distance from the center of the cross-section to the outer edge of the layer and r_{i-1} is the distance from the center of the cross-section to the inner edge of the layer. About the y and z axes, E_i becomes the elastic modulus and I_{i-1} becomes the moment of area of the layer. The moment of area of a full circle is

$$I_{i-1} = \frac{\pi}{4} R_{i-1}^4 \ , \tag{5-14}$$

and for a hollow ring is

$$I_{i-1} = \frac{\pi}{4}(R_{i-1}^4 - r_{i-1}^4) \ . \tag{5-15}$$

Figure 5-8 illustrates the longitudinal and radial growth of axes that are subject to gravity as well as different tropic influences.

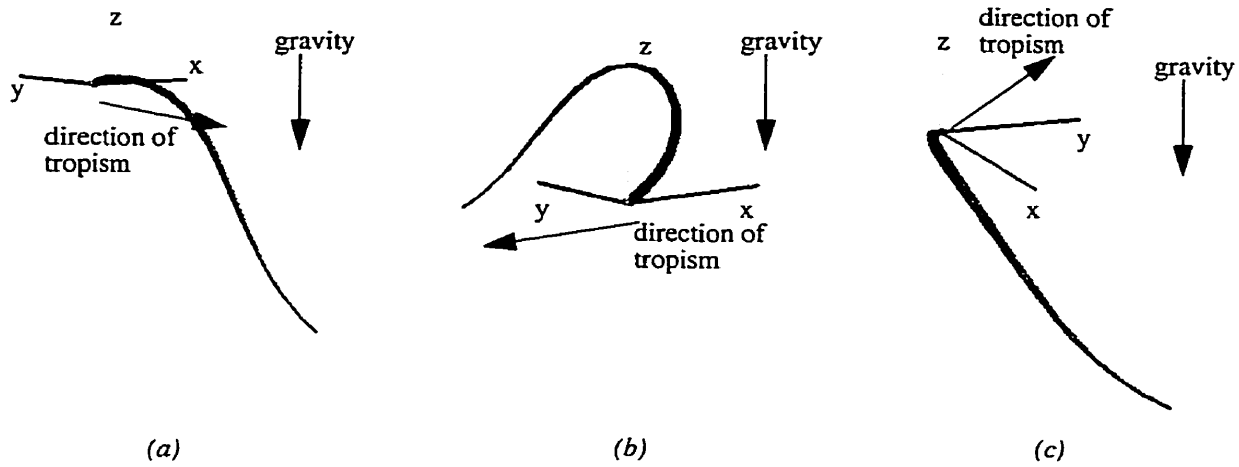


Figure 5-8: Simulations of longitudinal and radial growth of axes that are subject to different tropic influences. (a) An axis subject to a tropism in the negative y direction. (b) An axis subject to a strong tropism in the negative x direction. (c) An axis subject to a tropism that consists of positive x, y, and z components.

5.4 Branching Structures

The biomechanical model of the primary and secondary growth of an individual axis that is subject to external influences has been extended to developmental models of entire branching structures. A branching structure consists of a zero-order parent branch that supports higher-order axes (see Section 2.1.2). The divergence angle formed between neighboring child axes and the branching angle formed between a child axis and its parent branch are biological features inherent in a particular plant species. Aside from the initial orientation of a lateral axis, each axis within a branching structure develops independently of each other. That is, the shape and orientation of an axis is affected by the external forces and tropisms that are acting on the branching structure as a whole, and is not affected by the simultaneous development of other axes that are present in the structure. Therefore, the rotational displacement of each segment within an axis is calculated in the same manner as was described for individual axes (see Chapter 4 and the previous sections of this chapter).

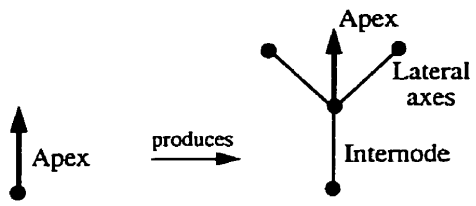


Figure 5-9: In a monopodial branching structure, an apex produces an internode followed by another apex, and one or more lateral branches.

5.4.1 Examples of Different Branching Structures

5.4.1.1 Monopodial Branching Structures

Monopodial branching structures have a single, well developed main axis that is parent to several child axes [41]. During the development of a monopodial structure, an apex will produce an internode followed by another apex, and one or more lateral axes (see Figure 5-9).

Several examples of monopodial branching structures that incorporate the biomechanical model of axis shape have been developed. Each example illustrates different branching angles and divergence angles between individual axes. Figure 5-10a,b show examples of monopodial structures with parent branches that grow vertically upward. In this figure, lateral axes assume branching angles of 90° with respect to the parent branch. The child branches in Figure 5-10a are placed in a spiral phyllotactic pattern about the parent branch, whereas the child branches in Figure 5-10b are placed in a decussate pattern about the parent branch (see Section2.1.3). All axes are subject to the force of gravity as well as an upward tropism.

The structures in Figure 5-10c,d show examples of parent branches that initially assume a horizontal orientation, but eventually bend downward due to their own weight. Lateral axes assume branching angles of 60° with respect to the parent branches. The child axes in Figure 5-10c are placed in a phyllotactic pattern about the parent branch, whereas the child axes in Figure 5-10d are placed in a distichuous pattern about the parent branch. All

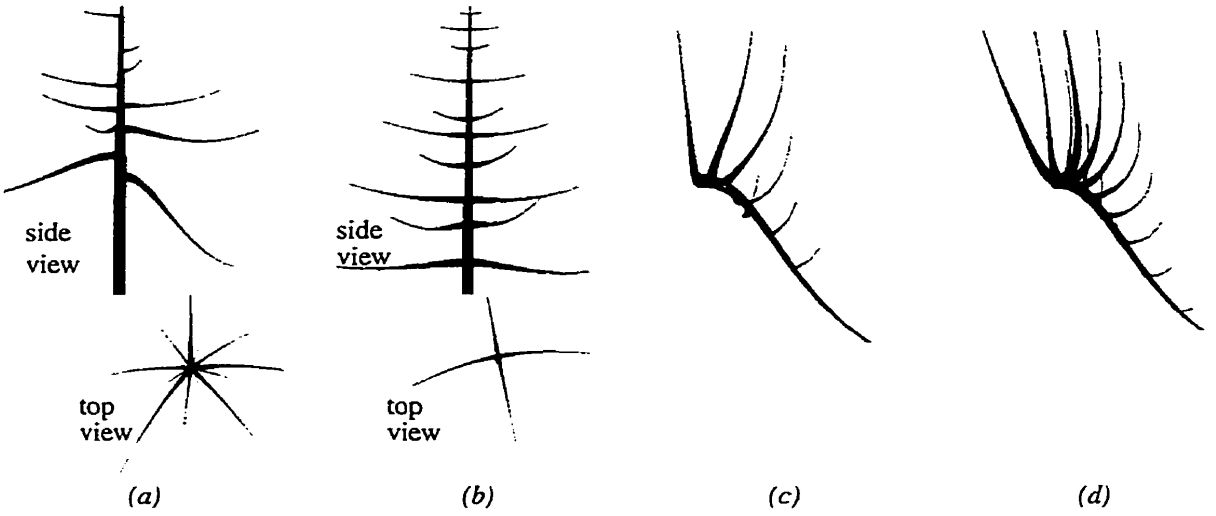


Figure 5-10: (a)-(b) Examples of monopodial structures with vertical parent branches. (a) The child branches are placed in a phyllotactic pattern about the parent branch. (b) The child branches are placed in a decussate pattern about the parent branch. (c)-(d) Examples of monopodial structures with parent branches that initially assume a horizontal orientation, but bend downward due to their own weight. (c) The child branches are placed in a phyllotactic pattern about the parent branch. (d) The child branches are placed in a distichuous pattern about the parent branch.

axes are subject to the force of gravity as well as an upward tropism; the parent branches are influenced by a weak tropism, whereas the child branches are influenced by a very strong tropism.

5.4.1.2 Polypodial Branching Structures

Polypodial branching structures consist of parent branches that may produce higher-order parent branches [41]. An apex will produce an internode followed by another apex, and possibly one or more lateral apices (see Figure 5-11).

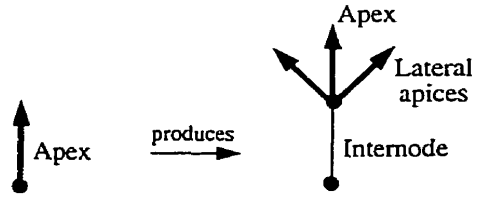
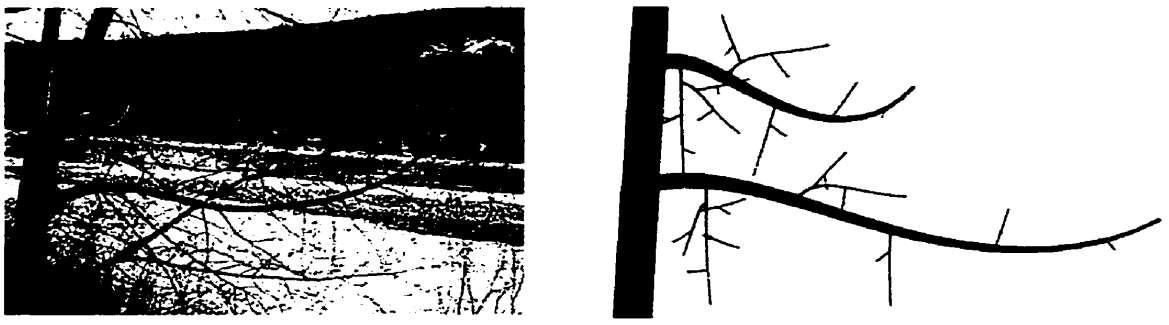


Figure 5-11: In a polypodial branching structure, an apex will produce an internode followed by another apex, and possibly one or more lateral apices.

(a) (b)

Figure 5-12: Examples of polypodial branching structures with lateral branches that are placed in a phyllotactic pattern about their respective parent branches. (a) The lowest ordered parent branch grows vertically upward. (b) The lowest ordered parent branch initially grows horizontally, but eventually bends downward due to its own weight.

A few examples of polypodial branching structures that incorporate the biomechanical model of axis shape have been developed. In Figure 5-12a, the lowest ordered parent branch grows vertically upward. In Figure 5-12b, the lowest ordered parent branch initially grows horizontally, but eventually bends downward due to its own weight. In both models, all lateral branches form a branching angle of 90° with respect to the parent branch, and are placed in a phyllotactic pattern about their respective parent branch. In Figure 5-13, a branching structure that is subject to the force of gravity and an upward tropism can be compared to a real photo of a tree branch.



(a) (b)

Figure 5-13: A branching structure that is subject to gravity and an upward tropism can be compared to a real photo of a tree branch.

Chapter 6

Implementation of a Biomechanical Model of Branch Shape Using L-Systems

This chapter discusses a method for implementing the growth of a plant axis that is subject to external forces. The implementation involves solving the equations that were presented in Chapter 4 that describe the mechanical properties of an axis subject to external forces. Primary growth, secondary growth, and tropisms are considered in the implementation as well. The developmental model of axis growth is implemented using the L-system methodology [41].

6.1 L-Systems and the Modeling Framework

L-systems provide a formally defined framework for plant modeling [24] [25], and have been used for image synthesis purposes [41]. The remainder of this section provides the rudiments of L-systems with turtle interpretation. The majority of the content in this section is quoted from [29] and [40].

An L-system is a parallel rewriting system operating on branching structures that are represented as strings of symbols with associated numerical parameters, called modules. The simulation of a branching structure begins with an initial string called the axiom, and proceeds in a sequence of discrete derivation steps. In each step, rewriting rules or produc-

tions replace all modules in the predecessor string by successor modules. The applicability of a production to a string depends on a predecessors context (in context-sensitive L-systems) and the values of parameters (in productions guarded by conditions). Typically, a production has the format:

```
id : lc < pred > rc : cond --> succ ,
```

where `id` is the production identifier (label), `lc`, `pred`, and `rc` are the left context, the strict predecessor, and the right context, `cond` is the condition, and `succ` is the successor. The strict predecessor and the successor are the only mandatory fields. For example, the L-system given below consists of axiom ω and two productions p_1 and p_2:

L-System 6-1

```
ω  : A(1)B(3)A(5)C(2)
p1 : A(x) --> A(x+1)
p2 : A(x) < B(y) > A(z) : y < 4 --> B(x+z)[A(y)] .
```

The production p_1 replaces module `A(x)` by `A(x+1)`. The context-sensitive production p_2 replaces a module `B(y)` with left context `A(x)` and right context `A(z)` by module `B(x+z)` supporting branch `A(y)`. The application of this production is guarded by condition `y<4`. Consequently, the first derivation step will be:

```
A(1)B(3)A(5)C(2) --> A(2)B(6)[A(3)]A(6)C(2) .
```

Production p_1 was applied to modules `A(1)` and `A(5)`. Production p_2 was applied to module `B(3)` because it occurred with the required left and right context, and the condition `3<4` is true. If no production applies to a module, then the module is replaced by itself.

Context-sensitive L-systems can be used to propagate a signal from one end of a string to the other. For example, to propagate a signal `m` from right to left, the following context-sensitive L-system can be used:

L-System 6-2

```
ω  : aaaaaam
p1 : a>m --> m
p2 : m --> a .
```

The first few strings generated by this L-system are:

```
aaaaaam
aaaaama
aaaamaa
aaamaaa .
```

A production using a left context can be used to propagate a signal from left to right.

Productions may include statements assigning values to local variables. These statements are enclosed in curly braces and are separated by semicolons. For example, the following L-system calculates the length of a vector represented by three components in the module V:

L-System 6-3

```
ω  : V(1,3,2)
p1 : V(x,y,z) : {l=sqrt(x*x+y*y+z*z);} --> f(l)
```

In this example, the result of the calculation is used in the successor to move the turtle forward by the calculated length. For details of the L-system syntax see [17], [38], and [41].

In contrast to the parallel application of productions in each derivation step, the interpretation of the resulting strings for image synthesis purposes proceeds sequentially. Reserved modules act as commands to a LOGO-style turtle [41]. At any point of the string, the turtle state is characterized by a position vector $\vec{P}$ and three mutually perpendicular orientation vectors $\vec{H}$, $\vec{U}$, and $\vec{L}$, indicating the turtle's heading, the up direction, and the direction to

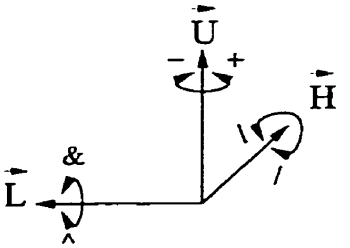


Figure 6-1: Controlling the turtle in three dimensions. Reproduced from [40].

the left (see Figure 6-1). Module F causes the turtle to draw a line in the current direction. Modules +, -, \, ^, / and \ rotate the turtle around one of the vectors $\vec{H}$, $\vec{U}$, or $\vec{L}$, as shown in Figure 6-1. The length of the line and the magnitude of the rotation angle can be given globally or specified as parameters of individual modules. The meaning of many symbols depends on the context in which they occur; for example, + and - denote arithmetic operators as well as modules that rotate the turtle.

Individual branches of a branching structure are enclosed by square brackets that appear in the string. During the interpretation of branches, the opening square bracket pushes the current position and orientation of the turtle on a stack, and the closing bracket restores the turtle to the position and orientation popped from the stack. To illustrate turtle interpretation of a string, an example of apical growth of a tree like branching structure is given by the following L-system:

L-System 6-4

```
ω  : FA
p1 : A --> [+(45)FB][-(45)FB]FA
p2 : B --> FB .
```

The strings produced in the first two derivation steps are:

```
FA
F[+(45)FB][-(45)FB]FA
F[+(45)FFB][-(45)FFB]F[+(45)FB][-(45)FB]FA .
```

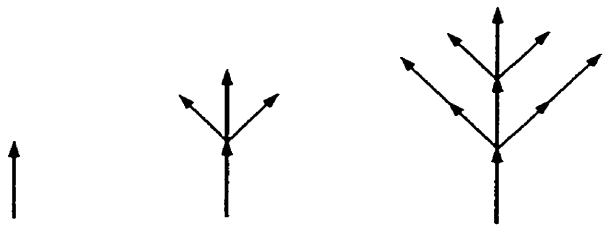


Figure 6-2: The turtle interpretation of the axiom and the first two strings produced by L-System 6-4.

Module A simulates the apex of the tree, which produces the main parent axis and left and right lateral axes. Module B simulates the growth of the lateral axes. The turtle interpretation of the axiom and strings resulting from the first two derivation steps is shown in Figure 6-2.

6.1.1 Environmentally-Sensitive L-Systems

The turtle interpretation of L-systems described above was designed to visualize models in a postprocessing step, with no effect on the L-system operation. However, position and orientation of the turtle are important when considering environmental phenomena, such as collisions with obstacles and exposure to light. The environmentally-sensitive extension of L-systems makes these attributes accessible during the rewriting process [40]. The generated string is interpreted after each derivation step, and turtle attributes found during the interpretation are returned as parameters to reserved query modules in the string. Each derivation step is performed as in parametric L-systems, except that the parameters associated with the query modules remain undefined. During the interpretation, these modules are assigned values that depend on the turtle's position and orientation in space. Syntactically, the query modules have the form `?X(x,y,z)`, where `X = P, H, U,` or `L`. Depending on the actual symbol `X`, the values of parameters `x`, `y`, and `z` represent a position or an orientation vector. In the two-dimensional case, the coordinate `z` may be omitted.

The operation of the query module is illustrated by a simple environmentally-sensitive L-system, given below:

L-System 6-5

ω : A

p_1 : A --> F(1)?P(x,y)-A

p_2 : F(k) --> F(k+1) .

The following strings are produced during the first three derivation steps:

μ'_0 : A

μ_0 : A

μ'_1 : F(1)?P(*,*)-A

μ_1 : F(1)?P(0,1)-A

μ'_2 : F(2)?P(*,*)-F(1)?P(*,*)-A

μ_2 : F(2)?P(0,2)-F(1)?P(1,2)-A

μ'_3 : F(3)?P(*,*)-F(2)?P(*,*)-F(1)?P(*,*)-A

μ_3 : F(3)?P(0,3)-F(2)?P(2,3)-F(1)?P(2,2)-A .

Strings μ'_0, μ'_1, μ'_2, and μ'_3 represent the axiom and the results of production application before the interpretation steps. Symbol * indicates an undefined parameter value in a query module. Strings μ_1, μ_2, and μ_3 represent the corresponding strings after interpretation. It has been assumed that the turtle is initially placed at the origin of the coordinate system, vector $\vec{H}$ is aligned with the y axis, vector $\vec{L}$ points in the negative direction of the x axis, and the angle of rotation associated with the module - is equal to 90°. Parameters of the query modules have values representing the positions of the turtle shown in Figure 6-3.

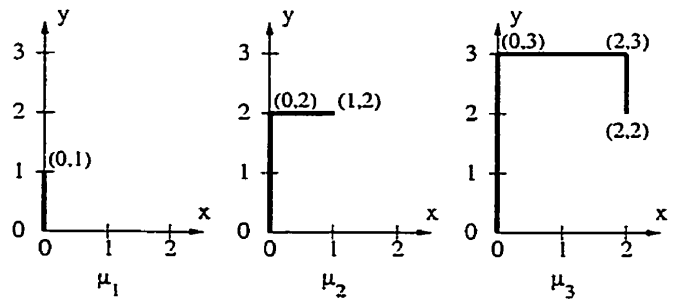


Figure 6-3: Assignment of values to query modules. Reproduced from [40].

6.1.2 Graphically-Defined Functions

The following description of graphically-defined functions is quoted from [37].

Values of parameters in a module may be set by function calls of the form `func(id,x)`. Several different functions may be used within an L-system, so the integer `id` is the function identifier. The real number `x` is the function argument. Functions are defined graphically using an interactive function editor (see Figure 6-4). The user specifies the function by manipulating its plot as a B-spline curve. The movement of each control point is constrained to the interval between the previous and the next control point along the x-axis. This guarantees that each x value will be mapped to exactly one y value, and the function will be unambiguously defined. The following example illustrates the use of a function call to set the value of a variable in a module `M`:

```
M(func(1,0)) .
```

A function with $id = 1$ is evaluated at $x = 0$. The resulting y that is returned from the function call sets the value of the parameter in the module. There is no limit to the number of functions that can be included in one model.

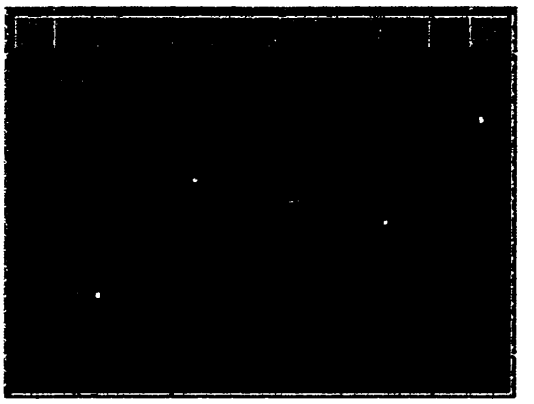


Figure 6-4: The interactive function editor.

6.2 Modeling The Growth of an Axis Consisting of Several Segments Using L-Systems

The theoretical model of a growing plant axis has been incorporated directly into the framework of L-systems. As a result, the system of equations that describes the biomechanical aspects of a plant becomes an inherent part of the model, and is automatically updated as the plant develops. Metaphorically speaking, the system of equations grows with the modeled plant. The L-system implementation makes it possible to produce diverse branching structures that incorporate the model of axis growth. In this section, the implementation of a longitudinally growing axis is presented.

6.2.1 Basic Simulation of the Growth of an Axis

The following example illustrates the basic form of an L-system that simulates the growth of an axis. Let module `f` represent a segment. The simulation of the growth of an axis begins with one segment. New segments are appended to the end of the axis at certain time intervals. A variable `T` is defined that specifies the number of iteration steps between the addition of new segments. A counter variable `t` is used to control when a new segment is added to the axis. At the beginning of the simulation, `t` is set to `0`. At each derivation step, `t` is incremented by `1`. If `t<T` then no new segment is added to the axis, whereas if `t=T` then a new segment is added, `t` is reset to `0`, and the simulation continues. The variable `t` is saved in a module `A(n,t)`. The parameter `n` specifies the number of the segment that was last added to the axis. At the beginning of the simulation, `n` is set to `i` for an axis that consists of i segments. When a new segment is added to the axis, `n` is decremented by `1`. When `n=0`, then the last segment of the axis has been added. The following L-system illustrates the growth of an axis consisting of six segments:

L-System 6-6

```
#define T 100
ω  : fA(5,0)
p1 : A(n,t) : t<T --> A(n,t+1)
p2 : A(n,t) : t==T && n>0 --> fA(n-1,0)
p3 : A(n,t) : t==T && n==0 --> * .
```

The axiom ω specifies that the simulation will begin with one segment. Production p_1 increments the time counter if `t<T`. Production p_2 adds a segment to the end of the axis when `t=T`, and resets the variables in module `A`. Production p_3 deletes module `A` when all of the segments have been added to the axis.

6.2.2 The Propagation of Information from One End of an Axis to the Other

In the physically-based model of an axis that consists of several segments, information is passed between segments in two directions. Specifically, positions and orientations of segments are accumulated from the beginning of the axis to the end. For example, segment $i-1$ is attached to the end of segment i and is oriented with respect to segment i. Torque and force are accumulated from the end of the axis to the beginning. For example, the torque about joint i depends on the torque about joint $i-1$ (see Section 4.1.3) [9].

The position and orientation of segment $i-1$ with respect to segment i is determined during the geometrical interpretation of the string. The string is geometrically interpreted from left to right, which propagates the position and orientation of the turtle from the beginning of the axis to the end.

Accumulating the torque and force from right to left, or from the end of the axis to the beginning, is performed using context-sensitive L-systems. To understand how informa-

tion can propagate from right to left, consider an axis that consists of several segments. From Section 4.1.3, we know that the torque about a joint i depends on the sum of the forces acting on segments $0 \ldots i-1$:

$$\vec{\tau}_{i_{external}} = \vec{r}_{i-1} \times \sum_{j=0}^{i-1} \vec{F}_j + \vec{\tau}_{i-1_{external}} . \tag{6-1}$$

Therefore, we would like to find the sum of the forces to the right of segment i:

$$F_{i_{accumulated}} = F_{i-1_{individual}} + F_{i-2_{individual}} + \cdots + F_{0_{individual}} \tag{6-2}$$

$$= F_{i-1_{individual}} + F_{i-1_{accumulated}} . \tag{6-3}$$

Two force values are associated with a segment $i-1$: an individual force $F_{i-1_{individual}}$ that is acting on the segment, and an accumulated force $F_{i-1_{accumulated}}$ that is the sum of the forces acting on segments $0 \ldots i-2$. To find the accumulated force to the right of segment i, a context-sensitive production is used to obtain the individual and accumulated force values from segment $i-1$.

The following L-system illustrates the propagation of the sum of the forces from right to left for an axis that consists of four segments:

L-System 6-7

```
ω  : f(3,0)f(1,0)f(4,0)f(5,0)
p1 : f(indiv,accum) > f(indiv_right,accum_right) -->
     f(indiv,indiv_right+accum_right)
```

Module `f(indiv,accum)` represents a segment. The individual force that is acting on a segment is represented by the variable `indiv`. The accumulated force to the right of the

segment is represented by the variable `accum`. The axiom consists of four segments, where the values for the individual forces acting on each segment are `3`, `1`, `4`, and `5`, from left to right, respectively. The accumulated force values are initially set to zero. The production p_1 calculates the value of the accumulated force to the right of the current segment. The information will be propagated from the end of the axis to the beginning over a period of three iterations. In general, it takes $m-1$ iterations to propagate information from right to left for an axis that consists of m segments. Figure 6-5 illustrates the accumulation of forces. Context-sensitive L-systems are also used to propagate torques from right to left, as specified by equation (6-1).

It is important to account for the propagation time of information from the end of the axis to the beginning. The delay in propagation of information does not affect the state of the axis when equilibrium has been reached.

6.2.3 Global Coordinate Systems vs. Local Coordinate Systems

In the implementation of the growth of an axis, the vector representing segment $i-1$ and the external force acting on the segment are represented in the global coordinate system (see Section 4.1.3.1). Therefore, the resulting torque is represented in the global coordinate system as well. To find the angular displacement of a segment, the torque vector is

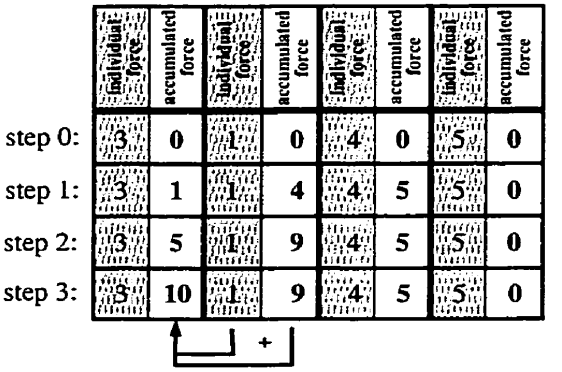


Figure 6-5: An illustration of how forces are accumulated from right to left over a period of three iteration steps for an axis that consists of four segments.

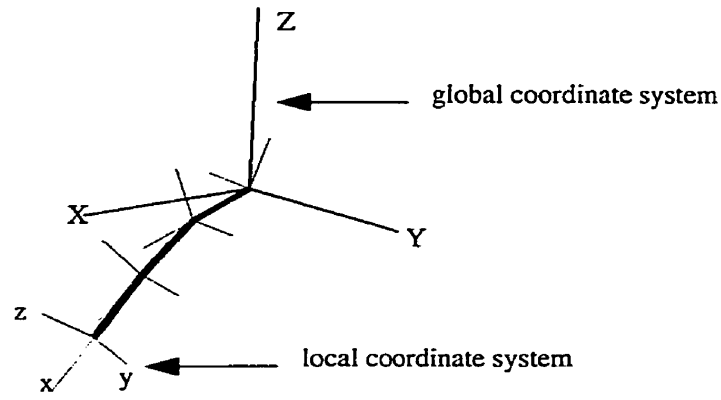


Figure 6-6: The global coordinate system is represented by (X,Y,Z), whereas the local coordinate system of a segment is represented by (x,y,z).

transformed such that it is represented in the local coordinate system of the segment (see Section 4.1.4). Since global and local coordinate systems are used, the following convention has been adopted in the L-system implementation: global coordinates are represented using capital letters `(X,Y,Z)`, whereas local coordinates are represented using lower case letters `(x,y,z)` (see Figure 6-6).

6.2.4 Representing a Joint and Segment of an Axis Using L-Systems

Joints and segments have several parameters associated with them. For example, each joint has three spring constants and each segment has a length and a mass. The values of these parameters must be retained throughout the simulation, therefore, L-system modules are defined that save information that is needed for each joint and segment.

Since the rotational displacement of a segment is assumed to be small, the angle of rotation of a segment is decomposed into three components. Each component specifies the angle of rotation about a local coordinate axis of the segment. Therefore, three L-system modules are defined that pertain to the three degrees of rotational freedom: the `/` symbol is associated with the rotation about the x-axis (or the $\vec{H}$ vector that is associated with the turtle), the `&` module is associated with the rotation about the y-axis (or the $\vec{L}$ vector), and the `-` module is associated with the rotation about the z-axis (or the $\vec{U}$ vector).

For each degree of freedom, other variables must be saved as well that are used throughout the simulation. These variables are saved as parameters that are defined within the /, &, and - modules. Five parameters for each module are defined: the angle of rotation a of segment $i-1$ about joint i, the angular velocity o, the torque t, the force F acting on segment $i-1$, and the sum of the forces f acting on segments $0 \ldots i-2$.

Two parameters that are not represented as three components are also saved: the mass m and length l of segment $i-1$. These parameters are saved in a module f. Therefore, four modules are used to describe a joint i and segment $i-1$ pair:

```
-(az,oz,tZ,FZ,fZ)&(ay,oy,tY,FY,fY)/(ax,ox,tX,FX,fX)f(l,m) .
```

6.2.5 Simulating the Growth of an Axis Using the Proposed Joint and Segment Representation

The following L-system simulates the growth of an axis, where each joint and segment pair is represented by the four modules that were introduced in the previous section:

L-System 6-8

```
#define FIX 1          /*function identification - individual force*/
#define FIY 2
#define FIZ 3
#define MASS 4         /*mass of segment*/
#define LENGTH 5       /*length of segment*/
#define NR 20          /*number of segments in the axis*/
#define T 100          /*iterations before a new segment is added*/

/****** Global Variables:
   step=used to get proper values from graphically-defined functions
******/

/****** Module definition:
A(n,t) -->
     n=segment number, controls when to stop adding segments
     t=time counter
```

```
-(az,oz,tZ,FZ,fZ) -->
     az=rotation angle about z-axis
     oz=angular velocity about z-axis
     tZ=Z component of torque
     FZ=force in Z direction acting on segment
     fZ=accumulated force from the end of the axis

f(l,m) -->
     l=length of segment
     m=mass of segment
******/

Start: {step = 1.0/NR;}

/*****begin with one joint
*****/
  ω:  +(90)\(90)                                    /*initial setup*/
      A(NR-1,0)
      -(0,0,0,func(FIZ,0),0)                      /*add first segment*/
      &(0,0,0,func(FIY,0),0)
      /(0,0,0,func(FIX,0),0)
      f(0,0)

  /*****depending on time t, either continue simulation, add a new
        segment, or stop simulation
  *****/
  p1: A(n,t) : t<T --> A(n,t+1)                     /*increment time*/

  p2: A(n,t) : t==T && n>0 -->                      /*add new segment*/
      -(0,0,0,func(FIZ,(NR-n)*step),0)
      &(0,0,0,func(FIY,(NR-n)*step),0)
      /(0,0,0,func(FIX,(NR-n)*step),0)
      f(func(LENGTH,(NR-n)*step),func(MASS,(NR-n)*step))
      A(n-1,0)

  p3: A(n,t) : t>=T && n==0 --> *     /*last segment has been added*/
```

The simulation of growth proceeds in the same manner as was discussed in Section 6.2.1. The simulation begins with one joint and segment, which are represented by the `-`, `&`, `/`, and `f` modules. No new modules are added while `t<T`. When `t=T` then a new joint and segment are added to the axis. The simulation of the growth of the axis stops when the last joint and segment have been added to the axis (when `n=0`).

The values of joint and segment parameters are set using graphically-defined functions (see Section 6.1.2). Functions are used to define values for the forces that are acting on the axis and the length and mass of each segment. The function is evaluated at $x = 0$ for the first segment in an axis. For each consecutive segment, a function is evaluated at $x = \frac{1.0}{NR}$, $x = \frac{2.0}{NR}, \ldots, x = 1.0$ for an axis that consists of NR segments. For example, a monotonically increasing function that defines the mass of each segment results in an axis with increasing segment masses from left to right.

6.2.6 Calculating the Torque About a Joint

In Section 4.1.3 it was shown that the torque $\vec{t}_{i_{external}}$ about a joint i is calculated by:

$$\vec{t}_{i_{external}} = \vec{r}_{i-1} \times \sum_{j=0}^{i-1} \vec{F}_j + \vec{t}_{i-1_{external}}, \tag{6-4}$$

where $\vec{r}_{i-1}$ is the vector that represents segment $i-1$ and $\vec{F}_j$ is the force acting on an individual segment. The corresponding L-system production that calculates the torque and angular displacement about a joint is:

```
?H(hX,hY,hZ)?U(uX,uY,uZ)?L(lX,lY,lZ) <
-(az,oz,tZ,FZ,fZ)&(ay,oy,tY,FY,fY)/(ax,ox,tX,FX,fX)f(l,m) >
?H(hXr,hYr,hZr)?U(uXr,uYr,uZr)?L(lXr,lYr,lZr)
-(azr,ozr,tZr,FZr,fZr)
&(ayr,oyr,tYr,FYr,fYr)
/(axr,oxr,tXr,FXr,fXr)
f(lr,mr) : *
{new_fX=FXr + fXr;                                  /*accumulate force*/
 new_fY=FYr + fYr;
 new_fZ=FZr + fZr;
 tX=(l*hYr*(FZ+new_fZ)-l*hZr*(FY+new_fY)) + tXr;/*calculate torque*/
 tY=(l*hZr*(FX+new_fX)-l*hXr*(FZ+new_fZ)) + tYr;
 tZ=(l*hXr*(FY+new_fY)-l*hYr*(FX+new_fX)) + tZr;
```

```
 Tx=tX*hX + tY*hY + tZ*hZ;            /*project onto local coordsys*/
 Ty=tX*lX + tY*lY + tZ*lZ;
 Tz=tX*uX + tY*uY + tZ*uZ;} -->
-(az+oz*dt,oz+((Tz-b*oz-func(EZ,(NR-n)*step)*az)/I)*dt,tZ,FZ,new_fZ)
&(ay+oy*dt,oy+((Ty-b*oy-func(EY,(NR-n)*step)*ay)/I)*dt,tY,FY,new_fY)
/(ax+ox*dt,ox+((Tx-b*ox-func(EX,(NR-n)*step)*ax)/I)*dt,tX,FX,new_fX)
f(l,m) .
```

The following discussion describes the different components of the production.

To calculate the torque about a joint i, the sum of the forces to the right of the joint must be determined. The accumulation of the forces is achieved by propagating information from the end of the axis to joint i, as discussed in Section 6.2.2. To obtain the sum of the forces that are to the right of joint i, the right context of the predecessor is applied:

```
-(az,oz,tZ,FZ,fZ)&(ay,oy,tY,FY,fY)/(ax,ox,tX,FX,fX)f(l,m) >
?H(hXr,hYr,hZr)?U(uXr,uYr,uZr)?L(lXr,lYr,lZr)
-(azr,ozr,tZr,FZr,fZr)
&(ayr,oyr,tYr,FYr,fYr)
/(axr,oxr,tXr,FXr,fXr)
f(lr,mr) ,
```

where all parameters that end in `r` refer to the parameters of joint $i-1$. The right context is used to obtain the sum of the forces that are applied to segments $0 \ldots i-2$ (see Section 6.2.2). The accumulation of forces is calculated by the equations

```
new_fX=FXr + fXr
new_fY=FYr + fYr
new_fZ=FZr + fZr ,
```

where `new_fX`, `new_fY`, and `new_fZ` are the x, y, and z components of the sum of the forces that are acting on segments $0 \ldots i-2$, respectively. The total force that is acting on segments $0 \ldots i-1$ is

$$\texttt{forceX}_{\texttt{total}}\texttt{=FX+new\_fX}$$
$$\texttt{forceY}_{\texttt{total}}\texttt{=FY+new\_fY}$$
$$\texttt{forceZ}_{\texttt{total}}\texttt{=FZ+new\_fZ}\ .$$

The components of the vector that represent segment $i-1$ must be determined in order to calculate the desired cross-product (see equation (6-4)). Assuming that the turtle's heading vector is parallel to the segment, the direction of the vector that represents segment $i-1$ can be found by querying for the turtle's heading vector using the module `?H(hXr,hYr,hZr)`. The components `hXr`, `hYr`, and `hZr` that are returned from the query are multiplied by the length `l` of segment $i-1$ to obtain the vector that represents segment $i-1$. The need for the `?U` and `?L` modules in the right context will become apparent in the next section.

The torque about joint i is computed by adding the result from the cross-product with the torque about joint $i-1$. The x, y, and z components of the torque are calculated by the equations

```
tX=(l*hYr*(FZ+new_fZ)-l*hZr*(FY+new_fY)) + tXr
tY=(l*hZr*(FX+new_fX)-l*hXr*(FZ+new_fZ)) + tYr
tZ=(l*hXr*(FY+new_fY)-l*hYr*(FX+new_fX)) + tZr .
```

6.2.7 Finding the Projection of the Torque Along Each Axis of a Segments Local Coordinate System

To find the angular displacement of a segment about a joint, the torque found in the previous section is projected onto the segments local coordinate axes (see Section 4.1.4). The $\vec{H}$, $\vec{L}$, and $\vec{U}$ vectors define the local x, y, and z axes of a segment, where $\vec{H}$ is assumed to be parallel to the previous segment, and $\vec{L}$ and $\vec{U}$ are principal axes that form a right-handed coordinate system. Therefore, the $\vec{H}$, $\vec{L}$, and $\vec{U}$ vectors associated with segment i are used to define the local coordinate system of segment $i-1$. The left context of the production in Section 6.2.6 is used to query for the $\vec{H}$, $\vec{L}$, and $\vec{U}$ vectors that define the local coordinate system of a segment:

```
?H(hX,hY,hZ)?U(uX,uY,uZ)?L(lX,lY,lZ) <
-(az,oz,tZ,FZ,fZ)&(ay,oy,tY,FY,fY)/(ax,ox,tX,FX,fX)f(l,m) >
?H(hXr,hYr,hZr)?U(uXr,uYr,uZr)?L(lXr,lYr,lZr)
-(azr,ozr,tZr,FZr,fZr)
&(ayr,oyr,tYr,FYr,fYr)
/(axr,oxr,tXr,FXr,fXr)
f(lr,mr) .
```

The projection of the torque $\vec{\tau}_i$ onto the local x-axis $\vec{H}_{i_x}$ is given by (see Section 4.1.4)

$$\vec{\tau}_{i_x} = \left(\frac{\vec{\tau}_i \cdot \vec{H}_{i_x}}{\|\vec{H}_{i_x}\|^2}\right)\vec{H}_{i_x} \ . \tag{6-5}$$

The projections of the torque onto the local y and z axes can be found in a similar manner. Only the dot product needs to be calculated to obtain the magnitudes of the projected torque components along the unit vectors of the coordinate axes. The projections of the torque onto the x, y, and z axes, respectively, are:

```
Tx=tX*hX + tY*hY + tZ*hZ
Ty=tX*lX + tY*lY + tZ*lZ
Tz=tX*uX + tY*uY + tZ*uZ ,
```

where Tx corresponds to τ_{i_x} which is the x component of the projection, and Ty and Tz correspond to the y and z components of the projection, respectively.

6.2.8 Calculating the Angular Velocity and the Angular Displacement of a Segment

The equilibrium equations that calculate the static state of an axis is solved iteratively using Euler's method (see Section 4.2). For example, the x component of the angular velocity ω_{i_x} of a segment about a joint i is (see Section 4.2.2)

$$\omega_{i_x}^{k+1} = \omega_{i_x}^{k} + \left(\frac{\tau_{i_x}}{I_{i_x}}\right)^{k} \Delta t \ , \tag{6-6}$$

where τ_{i_x} is the total torque about the x-axis (including the spring torque and the damping torque), I_{i_x} is the x component of the moment of inertia, k is the number of numerical iterations, and Δt is the time step. The equivalent statements that calculate the x, y, and z components of the angular velocity in the L-system are:

$$\begin{aligned} ox_{new} &= ox_{old}+((Tx-b*ox_{old}-kx*ax)/I)*dt \\ oy_{new} &= oy_{old}+((Ty-b*oy_{old}-ky*ay)/I)*dt \\ oz_{new} &= oz_{old}+((Tz-b*oz_{old}-kz*az)/I)*dt \ , \end{aligned}$$

where o_{old} is the angular velocity about a specific axis that was calculated in the previous derivation step, `a` is the angular displacement of the segment about an axis, `T` is the torque component due to external forces, `k` is the spring constant about an axis, `b` is the damping constant, `I` is the moment of inertia, and `dt` is the time increment.

The angular velocity is used to calculate the angle of rotation of a segment about its local x, y, and z axes. For example, the x component of the angle of rotation θ_{i_x} about a joint i is:

$$\theta_{i_x}^{k+1} = \theta_{i_x}^{k} + \omega_{i_x}^{k} \Delta t \ . \tag{6-7}$$

The equivalent L-system statements that calculate the x, y, and z components of the angle of rotation are:

$$\begin{aligned} ax_{new} &= ax_{old}+ox*dt \\ ay_{new} &= ay_{old}+oy*dt \\ az_{new} &= az_{old}+oz*dt \ . \end{aligned}$$

In L-systems, the magnitudes of the rotations that correspond to the /, &, and - modules are specified as the first parameter in the modules.The rotational displacements ax, ay, and az appear as the first parameters in the /, &, and - modules, respectively. Therefore, a segment is rotated about each local axis by the calculated angular displacements.

This completes the explanation of the production that was specified in Section 6.2.6.

6.3 Incorporating Tropisms into the Model

Section 5.2 discusses the method that was used to calculate the angle of rotation due to a tropism. In the implementation of the growth of an axis that is affected by a tropism, the simulation is split into three phases. In the first phase, the equilibrium state of an axis that is subject to external forces is computed. In the second phase, a new segment is added to the axis. In the third phase, the orientation of the newly added segment resulting from a tropism is calculated. The simulation of the growth of an axis that is affected by a tropism occurs as a cycle of the three phases.

The three phases occur at specific time intervals throughout the simulation. The state of equilibrium of an axis is calculated when t<T, and a new segment is added when t=T. The orientation of a segment due to a tropism is calculated when t>T and t<(T+Trop), where the constant Trop specifies the number of iterations that are used to calculate the angle of rotation of a segment due to a tropism.

Initially, a new segment is added such that it is parallel to the previous segment. In the L-system implementation, the new segment is initially represented by temporary modules that are used to calculate the displacement of the newly added segment due to a tropism. The displacement of a segment is calculated in an iterative fashion, as was discussed in Section 5.2. Once the angle of rotation of a newly added segment is calculated, the temporary modules are replaced by the regular modules that define a joint and segment pair (see Section 6.2.4).

The temporary modules that are used in the calculation of the displacement of a newly added segment are defined in the following L-system production:

```
?H(hX,hY,hZ)?U(uX,uY,uZ)?L(lX,lY,lZ) < A(n,t) : t==T && n>0 -->
@R(hX,hY,hZ,uX,uY,uZ,lX,lY,lZ)-(0,0)&(0,0)/(0,0)
f(func(LENGTH,(NR-n)*step))?H(0,0,0)A(n,t+1)
```

The module `A(n,t)` holds the value of the time counter and, therefore, regulates the three phases of the simulation (see Section 6.2.1). The parameters in the `@R` module set the $\vec{H}$, $\vec{L}$, and $\vec{U}$ vectors of the turtle. In the above production, the `@R` module is used to save the orientation of the turtle when the segment is added to the axis; the $\vec{H}$, $\vec{L}$, and $\vec{U}$ vectors define the local coordinate system of the newly added segment. The `-`, `&`, and `/` modules rotate the newly added segment about the local coordinate axes by the angular displacement due to the tropism. The first parameter in the rotation modules correspond to the angular displacement of the segment, and the second parameter corresponds to the angular velocity of the segment. The module `f` moves the turtle forward by the length of the segment. The `?H` module is used in the calculation of the torque due to the tropism.

The following L-system production calculates the torque and the angle of rotation due to a tropism:

```
@R(hX,hY,hZ,uX,uY,uZ,lX,lY,lZ)<
-(az,oz)&(ay,oy)/(ax,ox)f(l)?H(HX,HY,HZ)A(n,t) : t>T && t<(T+Trop)
{/*calculate torque*/
 tX=l*HY*func(TROPZ,(NR-n)*step) - l*HZ*func(TROPY,(NR-n)*step);
 tY=l*HZ*func(TROPX,(NR-n)*step) - l*HX*func(TROPZ,(NR-n)*step);
 tZ=l*HX*func(TROPY,(NR-n)*step) - l*HY*func(TROPX,(NR-n)*step);
 Tx=tX*hX + tY*hY + tZ*hZ;                /*project onto local axes*/
 Ty=tX*lX + tY*lY + tZ*lZ;
 Tz=tX*uX + tY*uY + tZ*uZ;} -->
 -(az+oz*dt,oz+((Tz-b*oz-func(EZ,(NR-n)*step)*az)/I)*dt)
 &(ay+oy*dt,oy+((Ty-b*oy-func(EY,(NR-n)*step)*ay)/I)*dt)
 /(ax+ox*dt,ox+((Tx-b*ox-func(EX,(NR-n)*step)*ax)/I)*dt)
 f(l)?H(0,0,0)A(n,t+1) .
```

The torque $\vec{\tau}_{i_{tropic}}$ about joint i that is a result of the tropism $\vec{F}_{tropic}$ that is acting on segment $i-1$ is (see Section 5.2)

$$\vec{\tau}_{i_{tropic}} = \vec{r}_{i-1} \times \vec{F}_{tropic} \ . \tag{6-8}$$

The torque is used to find the angle of rotation of a segment due to the tropism. The corresponding L-system equations calculate the torque due to a tropism:

```
tX=l*HY*FTZ - l*HZ*FTY
tY=l*HZ*FTX - l*HX*FTZ
tZ=l*HX*FTY - l*HY*FTX ,
```

where `(HX,HY,HZ)` is the unit vector that is parallel to the segment, `l` is the length of the segment, and `(FTX,FTY,FTZ)` is the tropism that is acting on the segment. In the L-system production, the tropism that is acting on the segment is specified using graphically-defined functions (see Section 6.1.2). The torque vector is projected onto the local coordinate axes of the newly added segment to obtain the magnitude of the torque about each axis (see Section 6.2.7):

```
Tx=tX*hX + tY*hY + tZ*hZ
Ty=tX*lX + tY*lY + tZ*lZ
Tz=tX*uX + tY*uY + tZ*uZ .
```

The rotational displacement of a segment is calculated by iteratively solving the equations

$$\omega_{i_x}^{k+1} = \omega_{i_x}^{k} + \left(\frac{\tau_{i_x}}{I_{i_x}}\right)^{k} \Delta t \ , \tag{6-9}$$

$$\theta_{i_x}^{k+1} = \theta_{i_x}^{k} + \omega_{i_x}^{k} \Delta t \ , \tag{6-10}$$

as was discussed in Section 6.2.8. The corresponding L-system equations are, respectively:

$ox_{new} = ox_{old}+((Tx-b*ox_{old}-kx*ax)/I)*dt$
$oy_{new} = oy_{old}+((Ty-b*oy_{old}-ky*ay)/I)*dt$
$oz_{new} = oz_{old}+((Tz-b*oz_{old}-kz*az)/I)*dt$,

$ax_{new} = ax_{old}+ox*dt$
$ay_{new} = ay_{old}+oy*dt$
$az_{new} = az_{old}+oz*dt$,

where o_{old} is the angular velocity about a specific axis that was calculated in the previous derivation step, `a` is the angular displacement of the segment about an axis, `T` is the torque about an axis, `k` is the spring constant about an axis, `b` is the damping constant, `I` is the moment of inertia, and `dt` is the time increment. The angle of rotation is calculated when `t>T` and `T<(T+Trop)`.

When `t=(T+Trop)` then the angle of rotation due to a tropism has been calculated and the temporary modules are replaced by the regular modules that define a joint and segment (see Section 6.2.4)[1]. The following L-system production replaces the temporary modules:

```
@R(hX,hY,hZ,uX,uY,uZ,lX,lY,lZ)-(az,oz)&(ay,oy)/(ax,ox)
f(l)?H(HX,HY,HZ)A(n,t) : t==(T+Trop) -->
-(az,0,0,func(FIZ,(NR-n)*step),0)
&(ay,0,0,func(FIY,(NR-n)*step),0)
/(ax,0,0,func(FIX,(NR-n)*step),0)
f(l,func(MASS,(NR-n)*step))
?H(0,0,0)?U(0,0,0)?L(0,0,0)A(n-1,0)
```

The segment assumes an orientation of `(ax,ay,az)` with respect to the segments local coordinate system, which is the resulting orientation due to the tropism. The `?H`, `?U`, and `?L` modules are needed for the left and right contexts in the production presented in Section 6.2.6. The variable `t` is reset to `0` so that the first phase of the simulation can restart.

---

1. The values of `T` and `Trop` are chosen such that a state of equilibrium of the axis is obtained within the allocated amount of time. Since the programmer will be able to visually see the axis during the simulation, he or she can choose an appropriate value that ensures that equilibrium of the axis is obtained.

6.4 Incorporating Radial Growth into the Model

Section 5.3 discusses the incorporation of radial growth into the model. The collection of pipes that make up a segment i is considered as a single layer that has an overall spring constant $\bar{\kappa}_i$ and rest angle $\bar{\phi}_i$ about each local axis. For example, the overall spring constant about the x-axis for a segment that consists of n layers is the sum of the individual spring constants of each layer (see equation (5-9)):

$$\bar{\kappa}_{i_{xn}} = \kappa_{i_{x1}} + \kappa_{i_{x2}} + \cdots + \kappa_{i_{xn}} = \bar{\kappa}_{i_{xn-1}} + \kappa_{i_{xn}} , \tag{6-11}$$

where $\bar{\kappa}_{i_{xn-1}}$ is the overall spring constant before the new layer was added. The overall rest angle about the x-axis of a segment with n layers is (see equation (5-10)):

$$\bar{\phi}_{i_{xn}} = \frac{\kappa_{i_{x1}}\phi_{i_{x1}} + \kappa_{i_{x2}}\phi_{i_{x2}} + \cdots + \kappa_{i_{xn}}\phi_{i_{xn}}}{\kappa_{i_{x1}} + \kappa_{i_{x2}} + \cdots + \kappa_{i_{xn}}} = \frac{\bar{\kappa}_{i_{xn-1}}\bar{\phi}_{i_{xn-1}} + \kappa_{i_{xn}}\phi_{i_{xn}}}{\bar{\kappa}_{i_{xn-1}} + \kappa_{i_{xn}}} , \tag{6-12}$$

where $\bar{\phi}_{i_{xn-1}}$ is the overall rest angle of the segment before the new layer was added. The modules that represent a joint and segment pair are expanded to retain the values of $\bar{\kappa}_{i_{xn-1}}$ and $\bar{\phi}_{i_{xn-1}}$ that are used in the calculations of $\bar{\kappa}_{i_{xn}}$ and $\bar{\phi}_{i_{xn}}$, respectively. Therefore, the expanded modules are

```
-(az,oz,tZ,AZ,fZ,kz,angz)
&(ay,oy,tY,AY,fY,ky,angy)
/(ax,ox,tX,AX,fX,kx,angx)
f(l,m) ,
```

where `k` is the overall spring constant about an axis and `ang` is the overall rest angle about an axis.

6.4.1 Calculating the Overall Spring Constant of a Newly Added Segment

The spring constant of a newly added segment is (see equation (5-11)):

$$\kappa_i = \frac{E_i I_{i-1}}{l_{i-1}}, \tag{6-13}$$

where l_{i-1} is the length of segment $i-1$, and E_i is the shear modulus about the x-axis at joint i, or the elastic modulus about the y and z axes. Assuming that the cross-section is circular, then the value for I_{i-1} is

$$I_{i-1} = \frac{\pi}{2} R_{i-1}^4 \tag{6-14}$$

about the x-axis and

$$I_{i-1} = \frac{\pi}{4} R_{i-1}^4 \tag{6-15}$$

about the y and z axes, where R_{i-1} is the radius of segment $i-1$.

The spring constant for a newly added segment is calculated when the segment is added to the end of the axis. As discussed in Section 6.3, a new segment is added by the following expanded production:

```
@R(hX,hY,hZ,uX,uY,uZ,lX,lY,lZ)-(az,oz)&(ay,oy)/(ax,ox)
f(l)?H(HX,HY,HZ)A(n,t) : t==(T+Trop)
{kx=(func(EX,(NR-n)*step)*PI*(w*w*w*w/16))/(2*l);
 ky=(func(EY,(NR-n)*step)*PI*(w*w*w*w/16))/(4*l);
 kz=(func(EZ,(NR-n)*step)*PI*(w*w*w*w/16))/(4*l);} -->
-(az,0,0,func(FIZ,(NR-n)*step),0,kz,az)
&(ay,0,0,func(FIY,(NR-n)*step),0,ky,ay)
```

```
/(ax,0,0,func(FIX,(NR-n)*step),0,kx,ax)
f(l,func(MASS,(NR-n)*step))
?H(0,0,0)?U(0,0,0)?L(0,0,0)A(n-1,0)  .
```

The equations

```
kx = (ex*PI*(w*w*w*w/16))/(2*l)
ky = (ey*PI*(w*w*w*w/16))/(4*l)
kz = (ez*PI*(w*w*w*w/16))/(4*l)
```

calculate the spring constant about each axis. Variable `e` is the shear modulus about the x-axis, or the elastic modulus about the y and z axes. Variable `w` is the initial width of the segment, `l` is the length of the segment, and `PI` is a constant. This is used in the calculation of the torque due to the spring.

6.4.2 Propagating a Signal to Indicate that a New Layer Has Been Added to an Axis

When a new segment is added to an axis, a signal is propagated from the end of the axis to the beginning, indicating that a new layer should be added to each segment. The propagation of information is achieved by checking to see if the width of the segment directly to the right of a segment i has increased. If so, the width of segment i should be increased and the calculations for the new rest angle and spring constant should be performed. Also, the module representing segment i should be set such that it propagates the signal to segment $i+1$.

The following production calculates the overall spring constant and rest angle for a segment that has grown radially. The production includes the propagation of a signal from right to left that specifies that a new layer should be added to a segment:

```
?H(hX,hY,hZ)?U(uX,uY,uZ)?L(lX,lY,lZ) <
-(az,oz,tZ,FZ,fZ,kz,angz)
&(ay,oy,tY,FY,fY,ky,angy)
/(ax,ox,tX,FX,fX,kx,angx)
f(l,m,w,dw) >
?H(hXr,hYr,hZr)?U(uXr,uYr,uZr)?L(lXr,lYr,lZr)
-(azr,ozr,tZr,FZr,fZr,kzr,angzr)
&(ayr,oyr,tYr,FYr,fYr,kyr,angyr)
/(axr,oxr,tXr,FXr,fXr,kxr,angxr)
f(lr,mr,wr,dwr) : *
{new_fX=FXr + fXr;                                  /*accumulate force*/
 new_fY=FYr + fYr;
 new_fZ=FZr + fZr;
 tX=(l*hYr*(FZ+new_fZ)-l*hZr*(FY+new_fY)) + tXr;/*calculate torque*/
 tY=(l*hZr*(FX+new_fX)-l*hXr*(FZ+new_fZ)) + tYr;
 tZ=(l*hXr*(FY+new_fY)-l*hYr*(FX+new_fX)) + tZr;
 Tx=tX*hX + tY*hY + tZ*hZ;            /*project onto local coordsys*/
 Ty=tX*lX + tY*lY + tZ*lZ;
 Tz=tX*uX + tY*uY + tZ*uZ;
 if(dwr > 0){                                        /*increment width*/
   new_kx=(func(EX,(NR-n)*step)*PI*(((w+dwr)^4-w^4)/16)))/(2*l);
   new_ky=(func(EY,(NR-n)*step)*PI*(((w+dwr)^4-w^4)/16)))/(4*l);
   new_kz=(func(EZ,(NR-n)*step)*PI*(((w+dwr)^4-w^4)/16)))/(4*l);
   angx=(kx*angx+new_kx*ax)/(kx+new_kx);       /*overall rest angle*/
   angy=(ky*angy+new_ky*ay)/(ky+new_ky);
   angz=(kz*angz+new_kz*az)/(kz+new_kz);
   kx=kx+new_kx;                            /*overall spring constant*/
   ky=ky+new_ky;
   kz=kz+new_kz;
   w=w+dwr;                                         /*increment width*/
}} -->
-(az+oz*dt,oz+((Tz-b*oz-kz*(az-angz))/I)*dt,tZ,FZ,new_fZ,kz,angz)
&(ay+oy*dt,oy+((Ty-b*oy-ky*(ay-angy))/I)*dt,tY,FY,new_fY,ky,angy)
/(ax+ox*dt,ox+((Tx-b*ox-kx*(ax-angx))/I)*dt,tX,FX,new_fX,kx,angx)
f(l,m,w,dwr)
```

The module `f` is expanded to retain information about a segments width `w`:

```
f(l,m,w,dw) .
```

The variable `dw` is used to propagate a signal to the left that indicates whether a new layer should be added to each segment. Initially, `dw` is set to `0`. The variable `dwr` that occurs in the right context of the predecessor is checked to see if the width of the segment to the right of the current segment has been increased. If `dwr=0` then the width of the right segment has not been increased, which specifies that a new segment has not been added to the

axis. If `dwr>0` then a new layer has been added to the segment to the right, indicating that a new layer should be added to the current segment. If a new layer is added to a segment, then a new rest angle and spring constant will be calculated, and the width of the current segment will be incremented by the value of `dwr`:

```
w=w+dwr ,
```

where `w` is the current width of the segment. To propagate the signal to the left, the width increment of the current segment is set to the width increment of the right segment:

```
dw = dwr .
```

This ensures that the segment that is directly to the left of the current segment will obtain the signal to increment its width. After the assignment, variable `dwr` is set to `0` so that new layers are not repeatedly added to the current segment.

6.4.3 Calculating the Overall Spring Constant of a Segment that Grew Radially

The spring constant for a segment is recalculated if the segment grew radially. If a segment grows radially, then the spring constant for the new layer is (see Section 5.3.1)

$$\kappa_i = \frac{E_i I_{i-1}}{l_{i-1}} , \tag{6-16}$$

where

$$I_{i-1} = \frac{\pi}{2}(R_{i-1}^4 - r_{i-1}^4) \tag{6-17}$$

about the x-axis and where

$$I_{i-1} = \frac{\pi}{4}(R_{i-1}^4 - r_{i-1}^4) \tag{6-18}$$

about the y and z axes. R_{i-1} is the distance from the center of the cross-section to the outer edge of the layer and r_{i-1} is the distance from the center of the cross-section to the inner edge of the layer. The corresponding L-system equations are:

```
new_kx=(ex*PI*(((w+dwr)^4-w^4)/16)))/(2*1)
new_ky=(ey*PI*(((w+dwr)^4-w^4)/16)))/(4*1)
new_kz=(ez*PI*(((w+dwr)^4-w^4)/16)))/(4*1) ,
```

where `dwr` is the width of the new layer.

The overall spring constant is specified by equation (6-11):

$$\bar{\kappa}_{i_{xn}} = \bar{\kappa}_{i_{xn-1}} + \kappa_{i_{xn}} . \tag{6-19}$$

The equivalent L-system equations are:

$$\begin{aligned} kx_{new} &= kx_{old} + new\_kx \\ ky_{new} &= ky_{old} + new\_ky \\ kz_{new} &= kz_{old} + new\_kz \ , \end{aligned}$$

where k_{old} is the spring constant of the segment before the new layer was added.

6.4.4 Calculating the Overall Rest Angle of a Segment

For a newly added segment that consists of one layer, the overall rest angle is the rotational displacement due to a tropism (see Section 5.3).

The overall rest angle of a segment that grows radially is calculated by equation (6-12):

$$\bar{\phi}_{i_n} = \frac{\bar{\kappa}_{i_{n-1}} \bar{\phi}_{i_{n-1}} + \kappa_{i_n} \phi_{i_n}}{\bar{\kappa}_{i_{n-1}} + \kappa_{i_n}} \quad . \tag{6-20}$$

The equivalent L-system equations are:

$$\begin{aligned} angx_{new} &= (kx_{old} * angx_{old} + new\_kx * ax) / (kx_{old} + new\_kx) \\ angy_{new} &= (ky_{old} * angy_{old} + new\_ky * ay) / (ky_{old} + new\_ky) \\ angz_{new} &= (kz_{old} * angz_{old} + new\_kz * az) / (kz_{old} + new\_kz) \quad , \end{aligned}$$

where ang_{old} is the overall rest angle of the segment before the new layer was added, a is the rest angle of the newly added layer, k_{old} is the spring constant before the new layer was added, and new_k is the spring constant of the new layer. The overall rest angle is used in the calculation of the torque due to the spring.

Chapter 7

Theory and Implementation of Collision Models

Often, as a plant develops, parts of the plant may collide with each other or with objects that are present in the environment. For example, leaves may collide with fruit, roots may collide with rocks, and axes may collide with other axes. In this chapter, we consider how collisions between elements of a plant axis and collisions between an axis and other objects in the environment can affect the shape and orientation of the axis.

The penalty method [35] is used to handle collisions between objects. When an intersection between two objects occurs, a force due to the collision is applied to each object, where the magnitude of the force depends on the amount of overlap between the colliding objects. The force resulting from the collision is incorporated into the equilibrium equations that govern the shape of a plant axis.

Obviously, a very large number of collisions can occur within a system. We have considered two special cases of collisions. The first case deals with collisions between plant elements, represented as spheres, that are attached to the end of individual axes. This method can be used to model collisions between fruit, such as cherries or grapes. The second case deals with collisions between an axis, represented as a cylinder, and a surface that is in the environment.

7.1 Detecting and Handling Collisions Between Spheres

Elements of plants, such as leaves or fruit, may collide with other elements of a plant. Elements, which are approximated by spheres, that are attached to the end of an axis are tested for collisions with other elements that are in the environment. When two or more elements intersect, a collision is detected and an appropriate force is applied to the elements such that they will rest against each other.

7.1.1 Detecting Collisions Between Two or More Spheres

A sphere is attached to the end of each axis that is present in the system. A collision is detected if two or more spheres intersect with each other. The case where a sphere intersects an axis is not dealt with. An intersection between two spheres occurs when the distance between the center of the spheres is less than the combined radii of the spheres. Each sphere is tested for an intersection with every other sphere that is in the environment.

7.1.2 Handling Collisions Using Newton's Third Law of Motion

Newton's third law of motion can be used to deal with collisions that occur between two or more spheres. Newton's third law of motion states that to every action there is always opposed an equal reaction [51]. Or, more formally, the force of object A exerted on object B is equal to the negative force of B exerted on A:

$$\vec{F}_{AB} = -\vec{F}_{BA} \ . \tag{7-1}$$

When two spheres collide, a force is exerted on each sphere in the direction of the line that intersects the two center of masses. That is, sphere A will exert a force on sphere B along the line of intersection, and sphere B will exert an equal but opposite force on sphere A along the line of intersection.

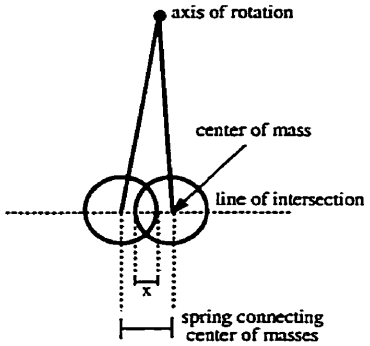


Figure 7-1: A spring is imagined to connect two intersecting spheres. At the point of collision, a force that is directed along the line of intersection is exerted on each sphere, where the magnitude of the force is given by Hooke's law.

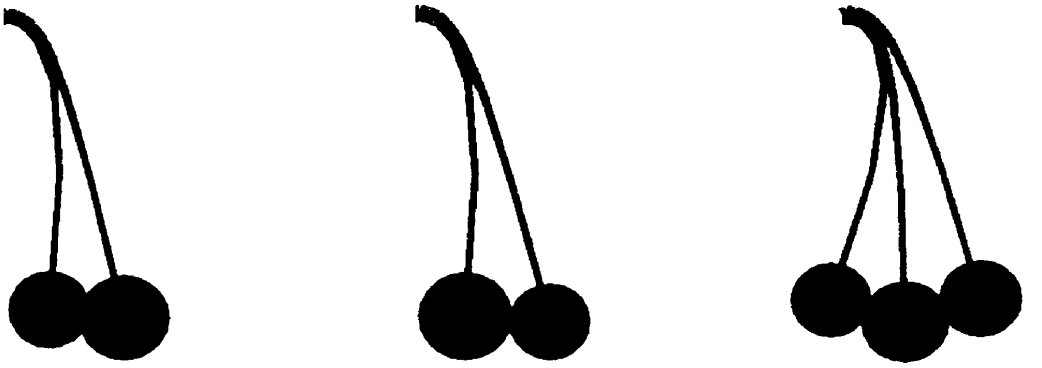
7.1.3 Finding the Magnitude of a Force Due to a Collision

If a collision occurs between two spheres, then a spring is imagined to connect the two intersecting spheres, where the ends of the spring are attached to each center of mass. The magnitude of the force that is exerted on each sphere is approximated using Hooke's law [51]:

$$F = -kx \ , \tag{7-2}$$

where k is the spring constant and x is the distance along the line of intersection that the spheres overlap (see Figure 7-1). A small spring constant specifies that the spring is quite flexible, which may result in overlapping spheres when a state of equilibrium has been obtained; a larger spring constant specifies that the spring is more rigid, which will ensure the spheres will not overlap when equilibrium has been reached (see Figure 7-2a,b).

Since the deformation of colliding spheres is not considered, equation (7-2) is used to specify the force that a sphere exerts on another sphere when they collide. If a sphere A collides with two or more spheres, then the total force that is exerted on A is the sum of all forces due to the collisions with the other spheres (see Figure 7-2c).



(a) (b) (c)

Figure 7-2: (a) The spring that connects the two spheres has a small spring constant. That is, the spring is quite flexible, so the spheres may overlap when equilibrium has been obtained. (b) The spring that connects the two spheres has a larger spring constant; the spring is more rigid so the spheres overlap less. (c) Three spheres have reached a state of equilibrium.

7.1.4 Finding a State of Equilibrium when Collisions Occur Between Spheres

Since a sphere is attached to the last segment of an axis, the force $\vec{F}_{0_{collision}}$ resulting from a collision between spheres is added to the force $\vec{F}_0$ due to gravity that is acting on the last segment of the axis:

$$\vec{F}_{0_{total}} = \vec{F}_0 + \vec{F}_{0_{collision}} \,. \tag{7-3}$$

Equation (7-3) is incorporated into equation (4-15), which calculates the torque about a joint i:

$$\vec{\tau}_{i_{external}} = \vec{r}_{i-1} \times \left(\vec{F}_{0_{total}} + \sum_{j=1}^{i-1} \vec{F}_j \right) + \vec{\tau}_{i-1_{external}} \,. \tag{7-4}$$

The final state of equilibrium of the axis is found using the equations that were presented in Section 4.2, where equation (7-4) is used as the torque due to external forces that are acting on the axis.

7.1.5 An Example of a Developmental Model of Cherries

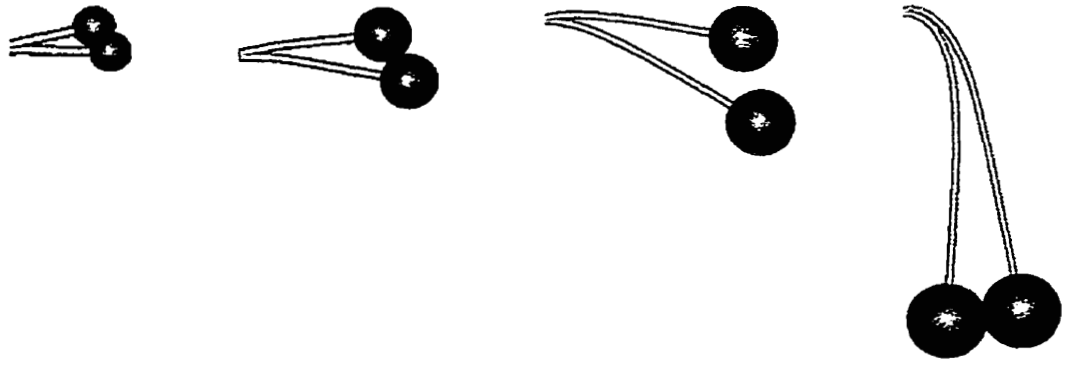
Figure 7-3 shows a biomechanical example of the growth of cherries. As the stems grow longitudinally, the cherries grow radially and get heavier. Over time, the stems bend downward due to the weight of the cherries.

At the start of the simulation of the growth of an axis with i segments, the cherry will be appended to segment $i-1$. As the axis grows, the cherry will be appended to each new segment that is added to the axis. Therefore, generalized versions of equations (7-3) and (7-4) are used in the simulation:

$$\vec{F}_{j_{total}} = \vec{F}_j + \vec{F}_{j_{collision}} , \tag{7-5}$$

$$\vec{\tau}_{i_{external}} = \vec{r}_{i-1} \times \sum_{j=0}^{i-1} \vec{F}_{j_{total}} + \vec{\tau}_{i-1_{external}} . \tag{7-6}$$

Collisions between cherries are detected throughout the simulation of the growth of the axes.



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

Figure 7-3: A developmental model of cherries. Collisions between cherries are detected throughout the development of the axes.

7.2 Detecting and Handling Collisions Between a Cylinder and a Plane

As an axis grows, it may interact with a surface such as the ground or a wall. When a segment of an axis collides with a surface, a collision is detected and an appropriate force is applied to the segment such that the segment will rest on the surface.

An axis consists of a circular cross-section that is swept along the skeleton of the axis, which is referred to as a generalized cylinder [12]. In this section, an intersection is detected between a cylinder and a plane. When an intersection occurs, an appropriate force is applied to the cylinder such that it will rest on the plane.

7.2.1 Detecting a Collision Between a Cylinder and a Plane

In this discussion, the end-point of a cylinder is defined to be the end of the skeletal component of the cylinder, and the end cross-section of the cylinder is defined to be the cross-section that contains the end-point.

If we are given the end-point of a cylinder and the radius of the cross-section at that point, then the position vector $\vec{p}$ of the point on the cross-section that is closest to the plane prior to a collision can be computed (see Figure 7-4). This point can then be used to detect whether the cylinder is above, on, or below the plane.

The normal of a plane can be used to calculate the closest point of a cylinder to the plane. Initially, the normal of the plane $\hat{n}_1$ is projected onto the normal of the end cross-section of the cylinder $\hat{n}_2$ (see Figure 7-5a,b):

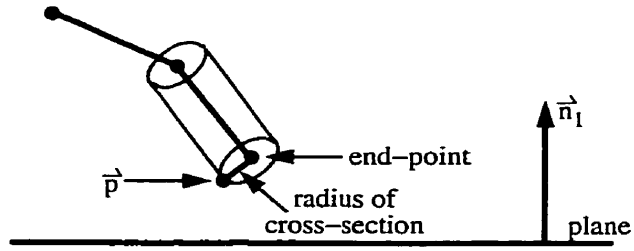


Figure 7-4: The position vector $\hat{p}$ of the point on the cross-section that is closest to the plane. $\hat{p}$ is used to detect if a collision occurs between the cylinder and the plane.

$$\hat{v}_1 = \frac{\hat{n}_1 \cdot \hat{n}_2}{\|\hat{n}_2\|^2}\hat{n}_2 \ . \tag{7-7}$$

The vector

$$\hat{v}_2 = \hat{v}_1 - \hat{n}_1 \tag{7-8}$$

specifies the line on which the closest point resides. The position vector $\hat{p}$ of the closest point to the plane is (see Figure 7-5c)

$$\hat{p} = r\frac{\hat{v}_2}{\|\hat{v}_2\|} + \hat{e} \ , \tag{7-9}$$

where r is the radius of the cross-section and $\hat{e}$ is the end-point of the cylinder.

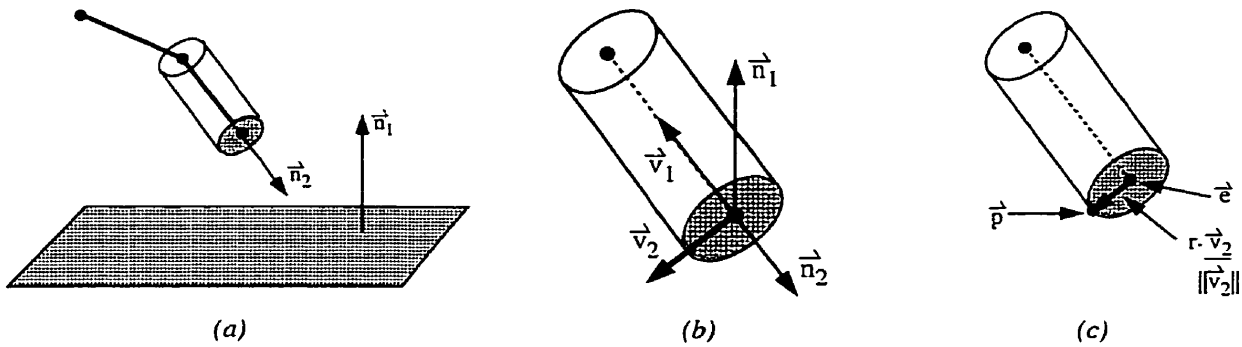


Figure 7-5: (a) The normal $\hat{n}_1$ of the plane and the normal $\hat{n}_2$ of the end cross-section, (b) The projection of $\hat{n}_1$ onto $\hat{n}_2$ to yield $\hat{v}_1$, (c) The closest point with position vector $\hat{p}$ of the cross-section to the surface.

The closest point of the cylinder to the plane is used to determine whether the cylinder lies above, on, or below the plane. This can be determined by using the vector equation of the plane [32]:

$$\hat{n}_1 \cdot (\hat{p} - \hat{p}_0) \ , \tag{7-10}$$

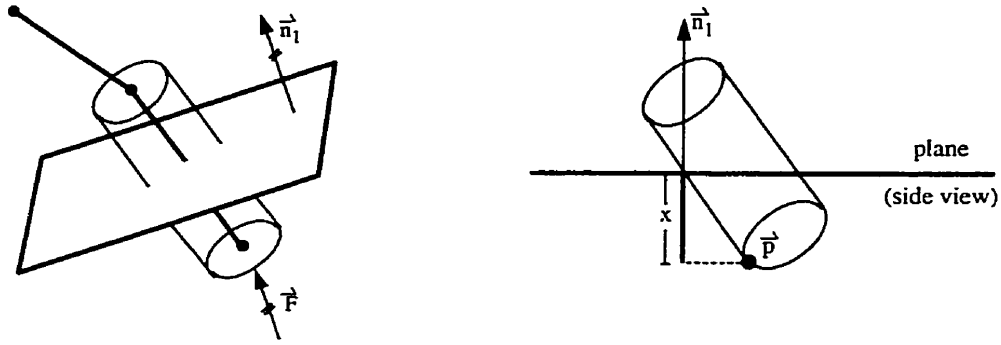
where $\hat{n}_1$ is the normal of the plane and $\hat{p}_0$ is the position vector of a point on the plane. The point with position vector $\hat{p}$ is on the plane if equation (7-10) results in a value of 0; if the equation results in a value that is greater than 0 then $\hat{p}$ is above the plane, whereas if the equation results in a value that is less than 0 then $\hat{p}$ is below the plane. Therefore, a collision of the cylinder with the plane occurs when equation (7-10) results in a value less than or equal to 0.

7.2.2 Finding the Magnitude of a Force Due to a Collision Between a Cylinder and a Plane

When a collision between a cylinder and a plane is detected, a force is exerted on the cylinder such that it lies on the plane. The plane exerts a force on the cylinder in the direction of the planes normal $\hat{n}_1$ (see Figure 7-6a). The magnitude of the force is approximated by Hooke's Law:

$$F = -kx \ , \tag{7-11}$$

where x is the shortest distance from the plane to the point with position vector $\hat{p}$. Given $\hat{p}$ and a point on the plane with position vector $\hat{p}_0$, then the shortest distance x can be calculated by finding the length of the projection of $\hat{p} - \hat{p}_0$ onto the normal $\hat{n}_1$ of the plane:



(a) *(b)*

Figure 7-6: (a) A force in the direction of the plane normal $\hat{n}_1$ is exerted on a protruding cylinder. (b) The shortest distance x from the plane to $\hat{p}$.

$$x = \frac{(\hat{p} - \hat{p}_0) \cdot \hat{n}_1}{\|\hat{n}_1\|} . \tag{7-12}$$

The spring constant k specifies the rigidity of the plane. A small k value specifies that the plane is quite elastic, which may result in intersections between the cylinder and the plane when a state of equilibrium has been obtained. A large k value specifies that the plane is quite rigid, which will ensure that the cylinder will lie on top of the plane.

7.2.3 Finding a State of Equilibrium when Collisions Occur Between a Cylinder and a Plane

A plant axis is approximated by a generalized cylinder. Each segment within an axis is tested for intersections with a surface, which is represented as a plane, that is present in the environment. The closest point of the end of each segment to the surface is used to detect whether collisions have occurred between the axis and the surface. If a segment intersects the surface, then the force due to the collision is computed and is added to the total external force that is acting on that segment (see Section 7.1.4). The external force that is acting on a segment is used in the calculation of the torque about a joint, which, in turn, is used to find the state of equilibrium of the axis. The final state of equilibrium of the axis is found using the equations of equilibrium that were presented in Section 4.2. Figure 7-7 illustrates the collisions of axes with horizontal and inclined surfaces.

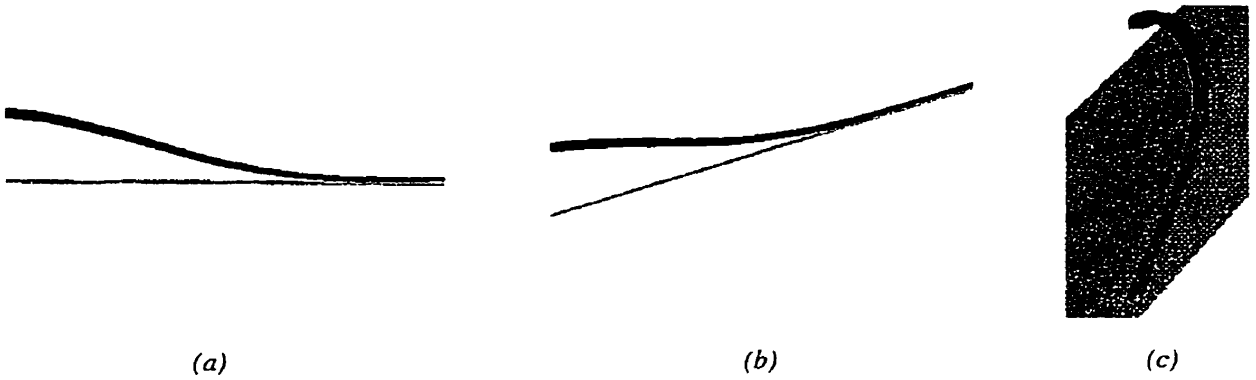


Figure 7-7: (a) An axis that collides with a horizontal plane. (b)-(c) Axes that collide with inclined planes.

7.2.4 An Example of a Developmental Model of a Plant Axis that Collides with a Surface

Figure 7-8 shows an example of the growth of an axis that collides with a surface. The axis is subject to a strong upward tropism, therefore, each newly added segment has a tendency to grow upward. Eventually, the axis starts to bend downward due to the force of gravity. Throughout the development of the axis, all segments are tested for collisions with the surface. If a segment intersects the surface, then a force is applied to the segment to keep it above the surface.

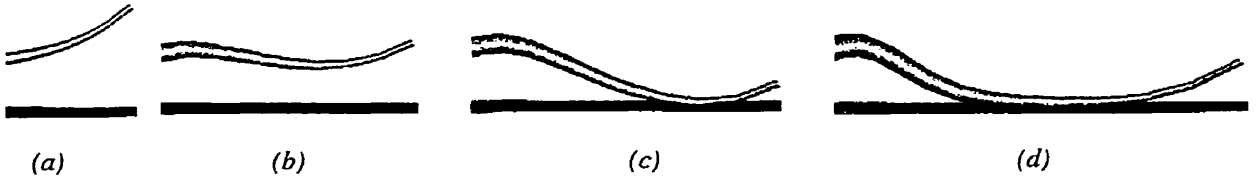


Figure 7-8: The growth of an axis that collides with a surface. The axis is subject to a strong upward tropism, as well as the force of gravity.

7.3 Implementation of the Collision Models Using Open L-Systems

7.3.1 Open L-Systems

Collisions between elements of an axis and collisions between an axis and a surface are implemented using Open L-systems. The following description of Open L-systems is quoted from [28] and [29].

The interaction of a plant with the environment can be conceptualized as two concurrent processes of information flow (see Figure 7-9). The plant process performs the following functions:

- reception of information about the environment in the form of scalar or vector values representing the stimuli perceived by specific organs,
- transport and processing of information inside the plant,
- generation of the response in the form of growth changes (e.g. development of new branches) and direct output of information to the environment (e.g. uptake and excretion of substances by a root tip).

Similarly, the environmental process includes mechanisms for the:

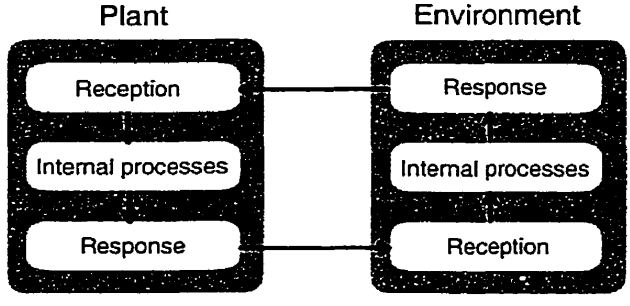


Figure 7-9: Conceptual model of plant and environment treated as communicating concurrent processes. Reproduced from [29].

- perception of the plant's actions.
- simulation of internal processes in the environment (e.g. the diffusion of substances or propagation of light).
- presentation of the modified environment in a form perceivable by the plant.

Open L-systems augment the functionality of environmentally-sensitive L-systems using a reserved symbol for bilateral communication with the environment. In short, parameters associated with an occurrence of the communication symbol can be set by the environment and transferred to the plant model, or set by the plant model and transferred to the environment. The environment is no longer represented by a simple function, but becomes an active process that may react to the information from the plant. Thus, plants are modeled as open cybernetic systems, sending information to and receiving information from the environment.

Communication modules of the form `?E(x1, ... , xm)` are used both to send and receive environmental information represented by the values of parameters `x1, ... , xm` (see Figure 7-10). To this end, the string resulting from a derivation step is scanned from left to right to determine the state of the turtle associated with each symbol. This phase is similar to the graphical interpretation of the string, except that the results need not be visualized. Upon encountering a communication symbol, the plant process creates and sends a message to the environment that includes all or a part of the following information:

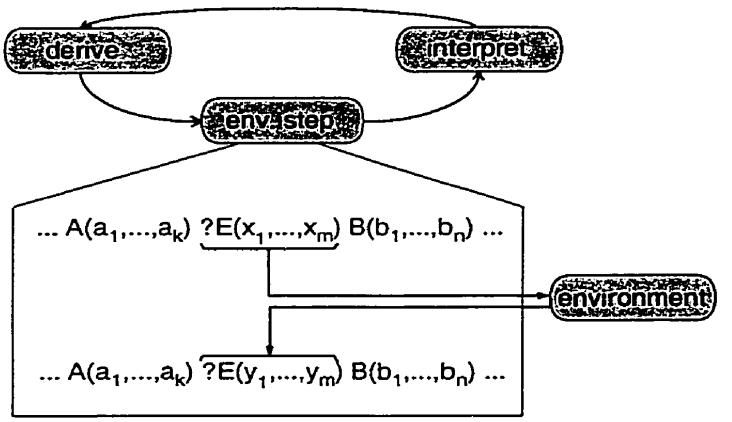


Figure 7-10: Information flow during the simulation of a plant interacting with the environment, implemented using an open L-system. Reproduced from [29].

- the address (position in the string) of the communication module (mandatory field needed to identify this module when a reply comes from the environment),
- values of parameters `xi`,
- the state of the turtle (coordinates of the position and orientation vector, as well as some other attributes, such as current line width),
- the type and parameters of the module following the communication module in the string.

The exact message format is defined in a communication specification file, shared between the programs modeling the plant and the environment. Consequently, it is possible to include only the information needed in a particular model in the messages sent to the environment. Transfer of the last message corresponding to the current scan of the string is signaled by a reserved end-of-transmission message, which may be used by the environmental process to start its operation.

The messages output by the plant modeling program are transferred to the process that simulates the environment using an interprocess communication mechanism provided by the underlying operating system (a pair of UNIX pipes, for example). The environment processes the information and returns the results to the plant model using messages in the following format:

- the address of the target communication module,
- values of parameters `yi` carrying the output from the environment.

The plant process uses the received information to set parameter values in the communication modules (see Figure 7-10). The use of addresses makes it possible to send replies only to selected communication modules. Details of the mapping of messages received by the plant process to the parameters of the communication modules are defined in the communication specification file.

After all replies generated by the environment have been received (a fact indicated by an end-of-transmission message sent by the environment), the next derivation step may be performed, initiating another cycle of the simulation.

Note that, by preceding every symbol in the string with a communication module it is possible to pass complete information about the model to the environment. Usually, however, only partial information about the state of a plant is needed as input to the environment. Proper placement of communication modules in the model, combined with careful selection of the information to be exchanged, provide a means for keeping the amount of transferred information at a manageable level.

The following simple example illustrates the operation of an open L-system. The model creates a branching structure consisting of straight line segments. The structure grows by adding a pair of segments to the end of existing branches unless a branch collides with another one. The occurrence of a collision is determined by the environment. To accomplish its task, the environment receives the information about the position of segments end points and tests whether two points occupy the same place or not.

The L-system model is as follows:

L-System 7-1

```
ω  : ?E(0)
p1 : ?E(c) : c==0 --> [+F/(180)?E(0)]F?E(0) .
```

The end point of a segment is represented by a communication module `?E` with one parameter. This parameter is initialized to `0`, and if the point collides with another point, the environment sets it to `1`. If the point does not collide, the parameter stays `0`. Production p_1 then creates two new branch segments only for points with parameter `0`.

The communication is set up such that with each communication module, the environment obtains its identification (the address in the string) and its position.

The first few steps of the simulation are described below.

Initialization:

The simulation begins with a single point `?E`. Before the first derivation step, the environmental step is performed and the environment receives the following information:

address: 0, ?E(0), position: 0,0,0 .

It is convenient to think of the address as the number of modules before the communication module. The position is equivalent to the initial position of the turtle. The point obviously does not collide with another point, thus the environment does not reply (i.e. sends an empty message) and the parameter of the module `?E` stays `0`.

Step 1:

The environmental step is followed by a derivation step, in which production p_1 is applied, replacing module `?E` with the string:

```
[+F/(180)?E(0)]F?E(0) ,
```

which is interpreted for visualization purposes (see Figure 7-11a). Now the environment receives two modules:

address: 4, ?E(0), position: 0.5,-0.866,0
address: 7, ?E(0), position: -0.5,-0.866,0 .

These two points do not collide and the environment again does not reply.

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

Figure 7-11: The first four steps produced by L-System 7-1. Reproduced from [28].

Step 2:

In the next derivation step, production p_1 is applied to both modules `?E`, resulting in the string:

```
[+F/(180)[+F/(180)?E(0)]F?E(0)]F[+F/(180)?E(0)]F?E(0) ,
```

which is visualized in Figure 7-11b. In the following environmental step, the environment receives four modules:

address: 8, ?E(0), position: 0, -1.7321, 0
address: 11, ?E(0), position: 1, -1.7321, 0
address: 18, ?E(0), position: 0, -1.7321, 0
address: 21, ?E(0), position: -1, -1.7321, 0 .

Since the first and third module occupy the same point, the environment returns a message in the form:

address: 8, ?E(1)
address: 18, ?E(1) .

The plant simulator receives this message and updates the parameters of the specified communication modules resulting in the string:

```
[+F/(180)[+F/(180)?E(1)]F?E(0)]F[+F/(180)?E(1)]F?E(0) .
```

The simulation then continues generating a branching structure which is similar to the Sierpinski gasket (see Figure 7-11).

7.3.2 Implementation of Collisions Between Spheres Using Open L-Systems

Collisions between spheres, which are used to model the interaction of fruit such as cherries and grapes, were discussed in Section 7.1. Open L-systems are used to handle collisions that may occur between two or more spheres. The environmental program detects whether a collision occurs and calculates the magnitude of the force that is exerted on each colliding sphere. The environmental program is listed in Appendix A.

A sphere is attached to the end of an axis by the following production:

```
A(n,t) : t==T && n==0 --> ?E(width/2,0,0,0)@O(width) .
```

The `@O(width)` draws a sphere with a width `width`. The communication module `?E(radius,forceX,forceY,forceZ)` is used to transfer information to and from the environmental program. The communication is set up such that with each communication module, the environment obtains the radius `radius` of the sphere and the position of the turtle. The environmental program uses this information to detect if a collision occurs between spheres that are present in the environment. If a collision occurs, the environmental program computes the force that results from the collision, which is approximated by Hooke's law (see Section 7.1.3):

$$F = -kx \ . \tag{7-13}$$

The components of the computed force are transferred from the environment to the plant model in the `forceX`, `forceY`, and `forceZ` parameters in the communication module. The force is added to the external force that is acting on the last segment of the axis. The following production adds the force that is returned in the `?E` module to the external force that is acting on the last segment:

```
?H(hX,hY,hZ)?U(uX,uY,uZ)?L(lX,lY,lZ) <
-(az,oz,tZ,FZ,fZ,kz,angz)
&(ay,oy,tY,FY,fY,ky,angy)
/(ax,ox,tX,FX,fX,kx,angx)
f(l,m,w,dw)?E(er,eX,eY,eZ) >
?H(hXr,hYr,hZr)?U(uXr,uYr,uZr)?L(lXr,lYr,lZr)
-(azr,ozr,tZr,FZr,fZr,kzr,angzr)
&(ayr,oyr,tYr,FYr,fYr,kyr,angyr)
/(axr,oxr,tXr,FXr,fXr,kxr,angxr)
f(lr,mr,wr,dwr) : *
{FX=FX+eX;                              /*add force due to collision*/
 FY=FY+eY;
 FZ=FZ+eZ;
 new_fX=FXr + fXr;                                /*accumulate force*/
 new_fY=FYr + fYr;
 new_fZ=FZr + fZr;
 tX=(l*hYr*(FZ+new_fZ)-l*hZr*(FY+new_fY)) + tXr;/*calculate torque*/
 tY=(l*hZr*(FX+new_fX)-l*hXr*(FZ+new_fZ)) + tYr;
 tZ=(l*hXr*(FY+new_fY)-l*hYr*(FX+new_fX)) + tZr;
 Tx=tX*hX + tY*hY + tZ*hZ;           /*project onto local coordsys*/
 Ty=tX*lX + tY*lY + tZ*lZ;
 Tz=tX*uX + tY*uY + tZ*uZ;
 if(dwr > 0){                                      /*increment width*/
   new_kx=(func(EX,(NR-n)*step)*PI*(((w+dwr)4-w4)/16)))/(2*l);
   new_ky=(func(EY,(NR-n)*step)*PI*(((w+dwr)4-w4)/16)))/(4*l);
   new_kz=(func(EZ,(NR-n)*step)*PI*(((w+dwr)4-w4)/16)))/(4*l);
   angx=(kx*angx+new_kx*ax)/(kx+new_kx);       /*overall rest angle*/
   angy=(ky*angy+new_ky*ay)/(ky+new_ky);
   angz=(kz*angz+new_kz*az)/(kz+new_kz);
   kx=kx+new_kx;                            /*overall spring constant*/
   ky=ky+new_ky;
   kz=kz+new_kz;
   w=w+dwr;                                         /*increment width*/
}} -->
-(az+oz*dt,oz+((Tz-b*oz-kz*(az-angz))/I)*dt,tZ,FZ,new_fZ,kz,angz)
&(ay+oy*dt,oy+((Ty-b*oy-ky*(ay-angy))/I)*dt,tY,FY,new_fY,ky,angy)
/(ax+ox*dt,ox+((Tx-b*ox-kx*(ax-angx))/I)*dt,tX,FX,new_fX,kx,angx)
f(l,m,w,dwr)?E(er,eX,eY,eZ)
```

The equations

```
FX=FX+eX
FY=FY+eY
FZ=FZ+eZ
```

add the force `(eX,eY,eZ)` that is returned from the environment to the external force `(FX,FY,FZ)` that is acting on the segment.

7.3.3 Implementation of Collisions Between an Axis and a Plane Using Open L-Systems

Collisions between an axis and a plane were discussed in Section 7.2. Open L-systems are used to handle collisions that may occur between a segment and a plane. The environmental program detects whether a collision occurs and calculates the magnitude of the force that is exerted on the segment due to the collision. The environmental program is listed in Appendix B.

To determine whether a collision between a segment and a plane has occurred, a communication module is associated with each segment within an axis:

```
-(az,oz,tZ,FZ,fZ,kz,angz)
&(ay,oy,tY,FY,fY,ky,angy)
/(ax,ox,tX,FX,fX,kx,angx)
?E(radius,length,forceX,forceY,forceZ)
f(l,m,w,dw) .
```

The communication module `?E(radius,length,forceX,forceY,forceZ)` is used to transfer information to and from the environmental program. The communication is set up such that with each communication module, the environment obtains the length `length` of the segment, the radius `radius` of the end cross-section of the segment, the position of the turtle, and the heading vector of the turtle. The environmental program uses this information to detect if a collision occurs between the segment and the plane. If a collision occurs, the environmental program computes the force that results from the collision, which is approximated by Hooke's law (see Section 7.2.2). The components of the computed force are transferred from the environment to the plant model in the `forceX`, `forceY`, and `forceZ` parameters in the communication module. The force due to the collision is added to the external force that is acting on the segment. The following production adds the force that is returned in the `?E` module to the external force that is acting on each segment:

```
?H(hX,hY,hZ)?U(uX,uY,uZ)?L(lX,lY,lZ) <
-(az,oz,tZ,FZ,fZ,kz,angz)
&(ay,oy,tY,FY,fY,ky,angy)
/(ax,ox,tX,FX,fX,kx,angx)
?E(er,el,eX,eY,eZ)
f(l,m,w,dw) >
?H(hXr,hYr,hZr)?U(uXr,uYr,uZr)?L(lXr,lYr,lZr)
-(azr,ozr,tZr,FZr,fZr,kzr,angzr)
&(ayr,oyr,tYr,FYr,fYr,kyr,angyr)
/(axr,oxr,tXr,FXr,fXr,kxr,angxr)
?E(err,elr,eXr,eYr,eZr)
f(lr,mr,wr,dwr) : *
{FX=FX+eX;                                /*add force due to collision*/
 FY=FY+eY;
 FZ=FZ+eZ;
 new_fX=FXr + fXr;                                 /*accumulate force*/
 new_fY=FYr + fYr;
 new_fZ=FZr + fZr;
 tX=(l*hYr*(FZ+new_fZ)-l*hZr*(FY+new_fY)) + tXr;/*calculate torque*/
 tY=(l*hZr*(FX+new_fX)-l*hXr*(FZ+new_fZ)) + tYr;
 tZ=(l*hXr*(FY+new_fY)-l*hYr*(FX+new_fX)) + tZr;
 Tx=tX*hX + tY*hY + tZ*hZ;            /*project onto local coordsys*/
 Ty=tX*lX + tY*lY + tZ*lZ;
 Tz=tX*uX + tY*uY + tZ*uZ;
 if(dwr > 0){                                          /*increment width*/
   new_kx=(func(EX,(NR-n)*step)*PI*(((w+dwr)4-w4)/16)))/(2*l);
   new_ky=(func(EY,(NR-n)*step)*PI*(((w+dwr)4-w4)/16)))/(4*l);
   new_kz=(func(EZ,(NR-n)*step)*PI*(((w+dwr)4-w4)/16)))/(4*l);
   angx=(kx*angx+new_kx*ax)/(kx+new_kx);       /*overall rest angle*/
   angy=(ky*angy+new_ky*ay)/(ky+new_ky);
   angz=(kz*angz+new_kz*az)/(kz+new_kz);
   kx=kx+new_kx;                             /*overall spring constant*/
   ky=ky+new_ky;
   kz=kz+new_kz;
   w=w+dwr;                                          /*increment width*/
}} -->
-(az+oz*dt,oz+((Tz-b*oz-kz*(az-angz))/I)*dt,tZ,FZ,new_fZ,kz,angz)
&(ay+oy*dt,oy+((Ty-b*oy-ky*(ay-angy))/I)*dt,tY,FY,new_fY,ky,angy)
/(ax+ox*dt,ox+((Tx-b*ox-kx*(ax-angx))/I)*dt,tX,FX,new_fX,kx,angx)
?E(er,el,eX,eY,eZ)f(l,m,w,dwr)
```

The equations

```
FX=FX+eX
FY=FY+eY
FZ=FZ+eZ
```

add the force `(eX,eY,eZ)` that is returned from the environment to the external force `(FX,FY,FZ)` that is acting on the segment.

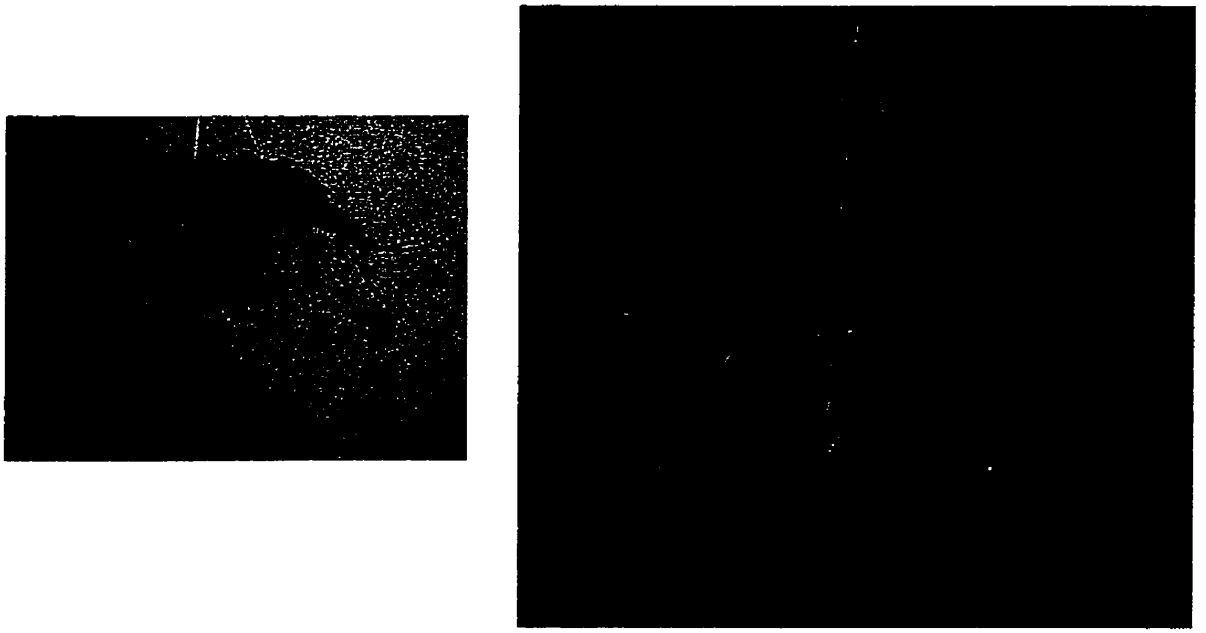
Chapter 8

Resulting Images

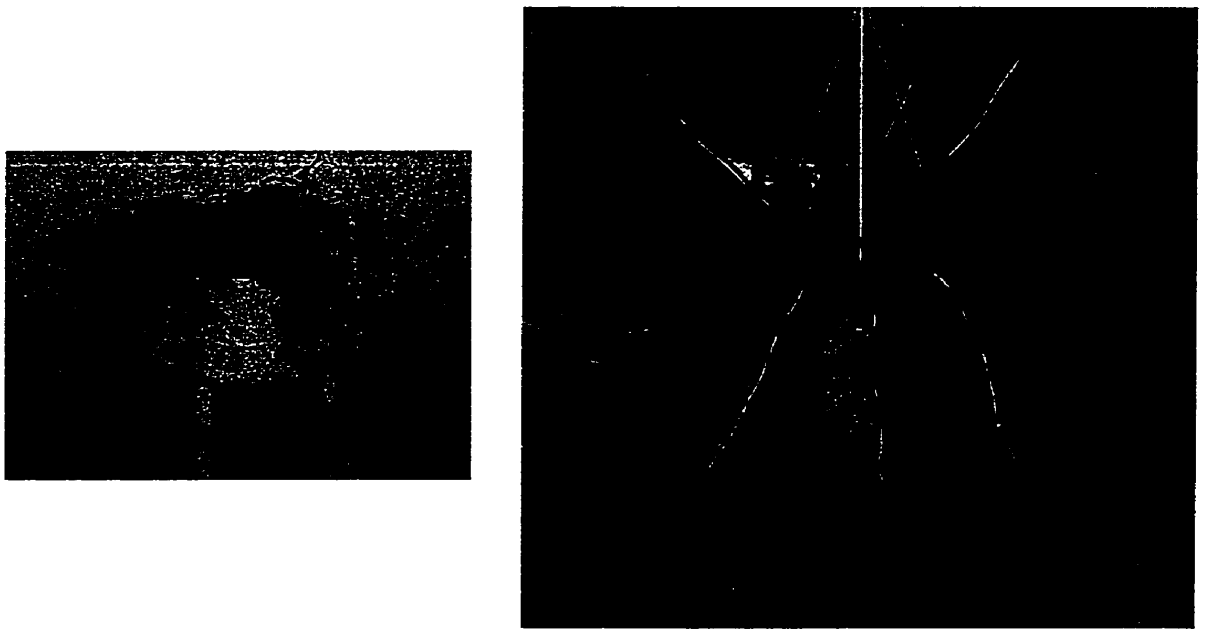
8.1 Examples of Plants

Several examples of plant models have been developed using L-systems that incorporate the biomechanical model of axis shape. Figure 8-1a and c show photographs of a hanging plant. The main axes have a tendency to grow in an upward direction, but eventually hang downward due to their own weight. The petioles are affected by a strong upward tropism. Figure 8-1b and d show the graphical models of the hanging plant.

Figure 8-2 shows images of a Spiraea bush. Figure 8-2a is a close-up photo of the inflorescence and their stems that are affected by a strong upward tropism. Figure 8-2c is a photo of an entire Spiraea bush. The parent branches of the stems initially assume a vertical orientation but begin to bend downward due to gravity. Figure 8-2b and d show the graphical models of the inflorescence and the bush, respectively.



(a) *(b)*



(c) *(d)*

Figure 8-1: (a), (c) Photos of a hanging plant. The branches bend downward due to their weight whereas the petioles are affected by a strong upward tropism. (b), (d) Graphical models of the plant.

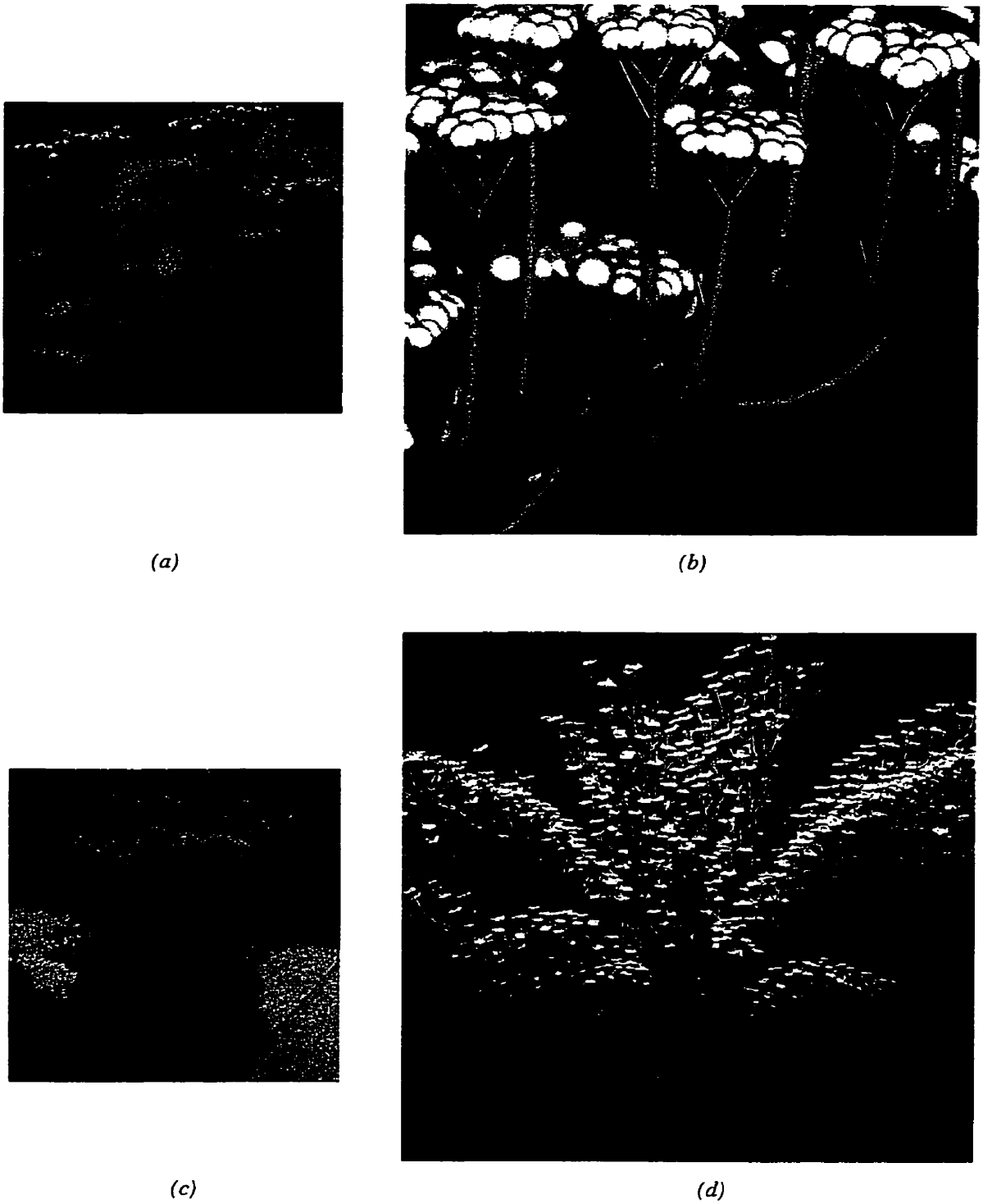


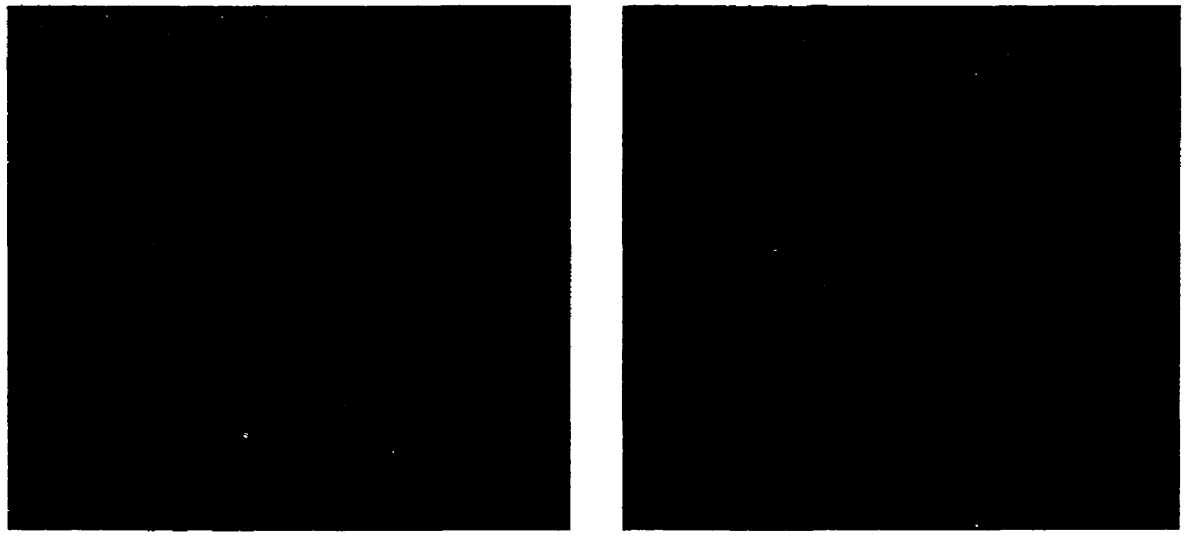
Figure 8-2: Images of a Spiraea bush. (a) A close-up photo of the inflorescence and their stems that are affected by a strong upward tropism. (c) An entire Spiraea bush. (b), (d) Graphical models of the inflorescence and the bush, respectively.

8.2 Examples of Fruit

The model that handles collisions between spheres was used to produce models of cherries and grapes. Figure 8-3a shows an example of cherries with stems that hang downward due to the weight of the fruit. Figure 8-3b shows an example of a grape vine that hangs downward due to gravity. The grapes and the cherries rest against each other in a state of equilibrium.

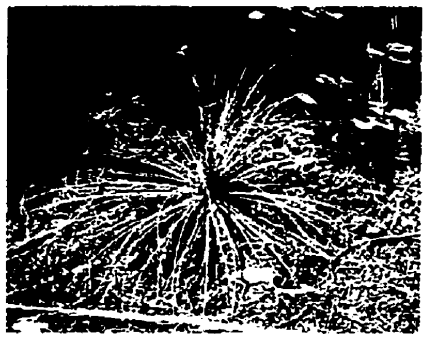
8.3 Examples of Axes Colliding with Surfaces

The model that handles collisions between axes and a plane was used to produce models of Bear grass and a Cycad. Figure 8-4 shows a photo and the corresponding graphical models of Bear grass that illustrate collisions that occur with the ground. Similarly, Figure 8-5 shows a photo and the corresponding graphical models of a Cycad with axes that collide with the ground.



(a) *(b)*

Figure 8-3: (a) A model of cherries. (b) A model of a grape vine that hangs downward due to gravity. The grapes and the cherries rest against each other in a state of equilibrium



(a)

(b)

(c)

Figure 8-4: (a) A photo of Bear grass that illustrates collisions that occur with the ground. (b), (c) Graphical models of Bear grass.

(a)

(b)

(c)

Figure 8-5: (a) A photo of a Cycad with axes that collide with the ground. (b), (c) Graphical models of a Cycad.

Chapter 9

Conclusion

9.1 Research Contributions

In this thesis, biomechanics was integrated into developmental plant models expressed using L-systems. The mechanical effects of external forces on an axis, coupled with longitudinal growth, radial growth, and tropisms, were incorporated into the developmental model, which was illustrated by examples of realistic plant images.

Elastic solid rods were used as an initial approximation to capture the effects of external forces on an axis. The rod model was extended to include biological factors, as was done by Fournier *et al.* [14]. The growth of an axis in a longitudinal direction was accomplished by adding segments to the apex of the axis. Radial growth of an axis was incorporated by adopting the pipe model that was introduced by Shinozaki *et al.* [45], which states that each segment within an axis consists of bundles of pipes that form outer layers on the segment. Tropisms were incorporated into the model by assuming that the orientation of each newly added segment is affected by a tropic influence, as was described by Digby and Firn [10]. The model of axis bending was further developed to incorporate collisions between fruit supported by axes and collisions between an axis and a mechanical object in the environment.

The biomechanical model of axis growth was implemented using the L-system methodology. This framework allowed for an individual axis to be implemented, which could then be reused several times to form entire branching structures.

The physical aspects of axis growth have never previously been implemented using the L-system methodology. Due to this fact, all previous L-system models did not incorporate the physical laws that govern the development of plant axes. The ability to incorporate biomechanics into L-system models allows for the creation of more realistic looking plant models. To this end, several illustrations of plant models that incorporate biomechanics were developed.

The biomechanical model that was developed can be used in multiple disciplines. Obviously, it can be used in computer graphics to produce animations and static images of plant models. It can also be used as simulations for biological research, and as models for landscape design.

9.2 Interpretation of Results and Future Work

An immediate extension of the work included in this thesis would be to refine the methods that are employed so that proper dynamics are used throughout the simulation. This extension could be based on the dynamical analysis of linkages, as discussed by Craig [9]. The proper dynamic solution could be used to solve the open problem of animating the motion of plants caused by external factors, such as wind. A repercussion of a proper dynamic solution is the equations that govern the dynamics of a plant axis may be more complicated, which, in turn, would result in a more complicated L-system implementation.

The issue of solving differential equations for two-point boundary value problems could be examined from a numerical methods perspective in more detail. To this end, the differential equations that describe the change in position of a continuous rod could be solved

using a more effective numerical method compared to the dynamic relaxation method that was used. Using sophisticated numerical methods may result in a more efficient means of obtaining the static state of an axis subject to external forces.

The model that was developed can be used to produce a variety of different plant species. As we have seen, a longitudinally growing axis that is subject to gravity will eventually hang downward due to the weight of the axis. But, in many instances, reaction wood is formed in certain species that opposes the effect of gravity; as an axis grows, the rest angles of all segments are altered so that the axis will not necessarily succumb to the force of gravity. This results in an axis that will maintain an upright or horizontal position. The effect of reaction wood can be incorporated into the existing model to increase biological faithfulness to models in nature.

Bibliography

[1] Bell, J. S., 1998. A Primer of Infinitesimal Analysis. Cambridge University Press.

[2] Bennet-Clark, T. A., & Ball, N. G. The Diageotropic Behavior of Rhizomes. *Journal of Experimental Botany*, 2(5): 169-203.

[3] Bennet-Clark, T. A., Younis, A. F., & Esnault, R., 1959. Geotropic Behavior of Roots. *Journal of Experimental Botany*, 10(28): 69-86.

[4] Bjorkman, T., 1988. Perception of Gravity by Plants. *Advances in Botanical Research*, 15: 1-41.

[5] Chiba, N., Ohkawa, S., Muraoka, K., & Miura, M., 1994. Visual Simulation of Botanical Trees Based on Virtual Heliotropism and Dormancy Break. *The Journal of Visualization and Computer Animation*, 5: 3-15.

[6] Chiba, N., & Ohshida, K., 1996. Visual Simulation of Leaf Arrangement and Autumn Colours. *The Journal of Visualization and Computer Animation*, 7: 79-93.

[7] Chiba, Y., 1991. Plant Form on the Pipe Model Theory. II. Quantitative Analysis of Ramification in Morphology. *Ecol. Res.*, 6: 21-28.

[8] Chou, P., & Pagano, N., 1967. Elasticity - Tensor, Dyadic, and Engineering Approaches. Dover Publications, Inc.

[9] Craig, J., 1989. Introduction to Robotics - Mechanics and Control, Second Edition. Addison-Wesley Publishing Company.

[10] Digby, J., & Firn, R. D., 1995. The Gravitropic Set-Point Angle (GSA): The Identification of an Important Developmentally Controlled Variable Governing Plant Architecture. *Plant, Cell and Environment*, 18: 1434–1440.

[11] Firn, R. D., & Digby, J., 1997. Solving the Puzzle of Gravitropism - Has a Lost Piece Been Found?. *Planta*, 203: S159-S163.

[12] Foley, J., van Dam, A., Feiner, S., & Hughes, J., 1996. Computer Graphics Principles and Practice, Second Edition in C. Addison-Wesley Publishing Company.

[13] Fourcaud, T., 1997. Relations Entre Croissance et Biomecanique de L'arbre. In *Modelisation et simulation de l'architecture des vegetaux*, Bouchon, J., de Reffeye, P., & Barthelemy, D. INRA, pp. 350-382.

[14] Fournier, M., Bailleres, H., & Chanson, B., 1994. Tree Biomechanics: Growth, Cumulative Prestresses, and Reorientations. *Biomimetics*, 2(3): 229-251.

[15] Fowles, G. R., & Cassiday, G. L., 1999. Analytical Mechanics, Sixth Edition. Saunders College Publishing.

[16] Goldstein, H., 1980. Classical Mechanics, Second Edition. London: Addison-Wesley Publishing Company.

[17] Hanan, J.S., 1992. Parametric L-Systems and Their Application to the Modeling and Visualization of Plants. Ph.D. Thesis, University of Regina.

[18] Hart, J. W., 1990. Plant Tropisms and Other Growth Movements. Unwin Hymann Ltd.

[19] Hejnowicz, Z., 1997. Graviresponses in Herbs and Trees: A Major Role for the Redistribution of Tissue and Growth Stresses. *Planta*, 203: S136-S146.

[20] Holton, M., 1994. Strands, Gravity and Botanical Tree Imagery. *Computer Graphics Forum*, 13(1): 57-67.

[21] Honda, H., Hatta, H., & Fisher, J., 1997. Branch Geometry in Cornus Kousa (Cornaceae): Computer Simulations. *American Journal of Botany*, 84(6): 745-755.

[22] Honda, H., Tomlinson, P. B., & Fisher, J. B., 1982. Two Geometrical Models of Branching of Botanical Trees. *Ann. Bot.*, 49: 1-11.

[23] Kohji, J., Yamamoto, R., & Masuda, Y., 1995. Gravitropic Response in Eichhornia Cressipes (Water Hyacinth) I. Process of Gravitropic Bending in the Peduncle. *J. Plant Res.*, 108: 387-393.

[24] Lindenmayer, A., 1968. Mathematical Models for Cellular Interaction in Development, Parts I and II. *Journal of Theoretical Biology*, 18: 280-315.

[25] Lindenmayer, A., 1971. Developmental Systems Without Cellular Interaction, Their Languages and Grammars. *Journal of Theoretical Biology*, 30:455-484.

[26] MacDonald, N., 1983. Trees and Networks in Biological Models. John Wiley & Sons Ltd.

[27] Marion, J., & Thornton, S., 1988. Classical Dynamics of Particles and Systems, Third Edition. Harcourt Brace Jovanovich, Publishers.

[28] Mech, R., 1998. CPFG Version 3.4 User's Manual. *Unpublished Manuscript.*

[29] Mech, R., & Prusinkiewicz, P., 1996. Visual Models of Plants Interacting with Their Environment. *Proceedings of SIGGRAPH '96, ACM SIGGRAPH*, pp. 397-410.

[30] Murray, C. D., 1927. A Relationship Between Circumference and Weight in Trees and its Bearing on Branching Angles. *J. Gen. Physiol.*, 10: 725-729.

[31] Myers, A.B., Firn, R.D., & Digby, J., 1994. Gravitropic Sign Reversal - A Fundamental Feature of the Gravitropic Perception or Response Mechanism in Some Plant Organs. *Journal of Experimental Botany*, 45: 77-83.

[32] Nicholson, W., 1995. Linear Algebra with Applications, Third Edition. PWS Publishing Company.

[33] Niklas, K. J., 1992. Plant Biomechanics - An Engineering Approach to Plant Form and Function. The University of Chicago Press Ltd.

[34] Papadrakakis, M., & Topping, B., 1999. Innovative Computational Methods for Structural Mechanics. Saxe-Coburg Publications.

[35] Platt, J., & Barr, A., 1988. Constraint Methods for Flexible Models. *Proceedings of SIGGRAPH '88, ACM SIGGRAPH*, pp. 279-288.

[36] Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P., 1992. Numerical Recipes in C 'The art of scientific programming', 2nd Edition. Cambridge University Press.

[37] Prusinkiewicz, P., Davidson, C., & Karwowski, R., 1999. Top Down Modeling of

Plants. *Unpublished Manuscript*, Department of Computer Science, University of Calgary.

[38] Prusinkiewicz, P., Hammel, M., Hanan, J., & Mech, R., 1996. Visual Models of Plant Development. In *Handbook of Formal Languages*, Rozenberg, G., & Salomaa, A. Springer-Verlag, Berlin.

[39] Prusinkiewicz, P., & Hanan, J., 1992. L-Systems: From Formalism to Programming Languages. In *Lindenmayer Systems: Impact on Theoretical Computer Science, Computer Graphics, and Developmental Biology*, Rozenberg, G., & Salomaa, A. Eds. Springer-Verlag, Berlin, pp. 193-211.

[40] Prusinkiewicz, P., James, M., & Mech, R., 1994. Synthetic Topiary. *Proceedings of SIGGRAPH '94, ACM SIGGRAPH*, pp. 351-358.

[41] Prusinkiewicz, P., & Lindenmayer, A., 1990. The Algorithmic Beauty of Plants. Springer-Verlag. With J.S. Hanan, F. D. Fracchia, D. R. Fowler, M. J. M. de Boer, and L. Mercer.

[42] Prusinkiewicz, P., Lindenmayer, A., & Hanan, J., 1988. Developmental Models of Herbaceous Plants for Computer Imagery Purposes. *Proceedings of SIGGRAPH '88, ACM SIGGRAPH*, pp. 141-150.

[43] Raven, P., Evert, R., & Eichhorn, S., 1992. Biology of Plants, Fifth Edition. Worth Publishers.

[44] Shinozaki, K., Yoda, K. H., & Kira, T., 1964. A Quantitative Analysis of Plant Form-The Pipe Model Theory. II. Further Evidence of the Theory and its Application in Forest Ecology. *Jap. J. Ecol.*, 14: 133-139.

[45] Shinozaki, K., Yoda, K., Hozumi, K., & Kira, T., 1964. A Quantitative Analysis of Plant Form-The Pipe Model Theory. I. Basic Analysis. *Jap. J. Ecol.*, 14: 97-105.

[46] Silk, W., 1984. Quantitative Descriptions of Development. *Ann. Rev. Plant Physiol.*, 35: 479-518.

[47] Stevens, P. S., 1974. Patterns in Nature. Little, Brown and Company.

[48] Strasburger, E., Noll, F., Schenck, H., & Karsten, G., 1908. A Text-Book of Botany. Macmillan & Company Ltd.

[49] Sugden, A., 1984. Longman Illustrated Dictionary of Botany. York Press.

[50] Suzuki, T., & Kitano, M., 1989. Lateral Bud Development and Shoot Growth in Low-Pruned *Morus Alba* Affected by Stem Orientation. *Physiologia Plantarum*, 76: 494-499.

[51] Young, H. D., 1992. University Physics, Eighth Edition. Addison-Wesley Publishing Company.

Appendix A

The environmental program that detects and handles collisions that occur between spheres (see Section 7.3.2) is based on the use of a library that takes care of the communication between a plant model and the environment. The communication library `comm_lib.h` that is used is provided by the L-system based plant modeling program CPFG [28]. In the following outline of the environmental program, it is assumed that the reader is familiar with the communication library, which is described in detail in [28].

The following environmental process detects and handles collisions that occur between two of more spheres:

Program A

```
#include <stdio.h>          /*include standard C libraries*/
#include <iostream.h>
#include <math.h>
#include "comm_lib.h"       /*include communication library*/
#include "vector.h"         /*include vector library*/

#define EPSILON 0.001       /*precision for comparisons*/
#define MAXQUERIES 1000     /*maximum number of queries*/

/***** Datastructure that stores all of the relevant information that
       is being transferred from the plant model to the environment.
*****/
struct item_type{
   float radius;            /*radius of sphere*/
   Vector centerOfMass;     /*center of sphere*/
   Vector collisionForce;   /*stores calculated force*/
   unsigned long id;        /*identifies communication module*/
   int master;              /*where info is received/sent*/
} queries[MAXQUERIES];
```

```
int num_queries;              /*number of queries that are sent*/

/***** Store the information received from cpfg into a datastructure
*****/
void StoreQuery(int master, unsigned long module_id,
                Cmodule_type *comm_symbol, CTURTLE *tu){

  if(tu->positionC < 3){           //error checking
    fprintf(stderr, "environment: turtle position missing.\n");
    return;
  }

  if(num_queries >= MAXQUERIES){   //error checking
    fprintf(stderr, "environment: too many queries!\n");
    return;
  }

  queries[num_queries].radius=comm_symbol[0].params[0].value;
  queries[num_queries].centerOfMass.setX(tu->position[0]);
  queries[num_queries].centerOfMass.setY(tu->position[1]);
  queries[num_queries].centerOfMass.setZ(tu->position[2]);
  queries[num_queries].id=module_id;
  queries[num_queries].master=master;

  num_queries++;
}

/***** Determine whether collisions have occurred between spheres
*****/
void DetermineResponse(void){
   Vector line;                  /*direction of computed force*/
   double force=0;               /*magnitude of force*/
   Vector localForce;            /*magnitude and direction of force*/
   double currentDistance;       /*distance between two spheres*/
   double touchDistance;         /*distance between centers of spheres*/
   double collisionSpring=20.0;  /*spring constant (Hooke's Law)*/
   Cmodule_type *comm_symbol;    /*holds info sent to plant model*/
   comm_symbol=new Cmodule_type[num_queries]; /*allocation of space*/

   for(int i=0; i<num_queries; i++){
      queries[i].collisionForce.set(0,0,0);/*initialize force to zero*/
      comm_symbol[i].num_params=4;   /*datastructure has 4 parameters*/
      for(int j=1; j<4; j++){       /*send back the last 3 parameters*/
         comm_symbol[i].params[j].set=1;
         comm_symbol[i].params[j].value=0;
      }
   }

   for(i=0; i<num_queries; i++){ /*compare this sphere with all others*/
      for(int j=i+1; j<num_queries; j++){              /*second sphere*/
         /***line of force***/
         line.sub(queries[i].centerOfMass, queries[j].centerOfMass);
```

```
            /***distance fruit are apart***/
            currentDistance=line.length();
            /***distance at which two spheres will touch***/
            touchDistance=queries[i].radius+queries[j].radius;

            if(currentDistance<=touchDistance){    /*if two spheres touch*/
               force=collisionSpring*(touchDistance-currentDistance);
               line.normalize(line);      /*normalize direction of force*/

               /***magnitude and direction of force***/
               localForce.scalarMult(force, line);
               queries[i].collisionForce.add(localForce,
                                             queries[i].collisionForce);
               localForce.scalarMult(-force, line);
               queries[j].collisionForce.add(localForce,
                                             queries[j].collisionForce);
            }
         }
      }

      for(i=0; i<num_queries; i++){       /*for each sphere, send force*/
         comm_symbol[i].params[1].value=queries[i].collisionForce.getX();
         comm_symbol[i].params[2].value=queries[i].collisionForce.getY();
         comm_symbol[i].params[3].value=queries[i].collisionForce.getZ();
         CSSendData(queries[i].master, queries[i].id, &comm_symbol[i]);
      }
   }

   /***** Controls the loop of data exchange
   *****/
   void MainLoop(void){
     Cmodule_type two_modules[2];
     unsigned long module_id;
     int master;
     CTURTLE turtle;

     for(;;){       //infinite loop - until signal 'exit' comes
       CSBeginTransmission();
       num_queries=0;

       /*process data*/
       while(CSGetData(&master, &module_id, two_modules, &turtle))
          StoreQuery(master, module_id, two_modules, &turtle);

       DetermineResponse();

       if(CSEndTransmission()) break;  //End transmission
     }
   }
```

```
/***** Main function
*****/
int main(int argc, char **argv){
  CSInitialize(&argc, &argv); //initialize the communication
  MainLoop();
  CTerminate();               //should be last function call
  return 1;
}
```

The functions `StoreQuery()`, `MainLoop()`, and `main()` take care of the communication process and are provided by the communication library `comm_lib.h`. The `StoreQuery()` function saves information that is sent from the plant model to the environment into the datastructure `queries[]`. Specifically, the radius of the sphere is saved in `queries[].radius` and the position of the turtle is saved in `queries[].centerOfMass`. The functions `MainLoop()` and `main()` control the loop of data exchange (see [28]).

The function `DetermineResponse()` detects any collisions that occur between spheres and computes the resulting forces. The variable `comm_symbol` of type `Cmodule_type` (which is defined in `comm_lib.h`) is used to transfer data from the environment to the plant model. The first `for-loop` in the function initializes `comm_symbol` so that the appropriate data will be returned to the plant model. The second nested `for-loop` detects whether a collision occurs between spheres and computes the forces that result from any occurring collisions. The resultant forces are stored in `queries[].collisionForce`. The last `for-loop` assigns the computed force values to `comm_symbol`. Function `CSSendData()` transfers the data that is stored in `comm_symbol` to the plant model.

Appendix B

The environmental program that detects and handles collisions that occur between a cylinder and a plane (see Section 7.3.3) is based on the use of a library that takes care of the communication between a plant model and the environment. The communication library `comm_lib.h` that is used is provided by the L-system based plant modeling program CPFG [28]. In the following outline of the environmental program, it is assumed that the reader is familiar with the communication library, which is described in detail in [28].

The following environmental process detects and handles collisions that occur between a cylinder and a plane:

Program B

```
#include <stdio.h>          /*include standard C libraries*/
#include <iostream.h>
#include <math.h>
#include "comm_lib.h"       /*include communication library*/
#include "vector.h"         /*include vector library*/

#define EPSILON 0.001       /*precision for comparisons*/
#define MAXQUERIES 1000     /*maximum number of queries*/

/***** Datastructure that stores all of the relevant information that
       is being transferred from the plant model to the environment.
*****/
struct item_type{
   float radius;          /*radius of segment*/
   float length;          /*length of segment*/
   Vector position;       /*position of turtle*/
   Vector heading;        /*heading vector of turtle*/
   Vector force;          /*stores calculated force*/
   unsigned long id;      /*identifies communication module*/
   int master;            /*where info is received/sent*/
```

```
} queries[MAXQUERIES];

int find_collision(item_type,Vector*,Vector,Vector);
int num_queries;

/***** Store the information received from the plant model into
       a datastructure
*****/
void StoreQuery(int master, unsigned long module_id,
                Cmodule_type *comm_symbol, CTURTLE *tu){

  if(tu->positionC < 3){           //error checking
    fprintf(stderr, "environment: turtle position missing.\n");
    return;
  }

  if(tu->headingC < 3){            //error checking
    fprintf(stderr, "environment: turtle heading missing.\n");
    return;
  }

  if(num_queries >= MAXQUERIES){   //error checking
    fprintf(stderr, "environment: too many queries!\n");
    return;
  }

  queries[num_queries].radius=comm_symbol[0].params[0].value;
  queries[num_queries].length=comm_symbol[0].params[1].value;
  queries[num_queries].position.setX(tu->position[0]);
  queries[num_queries].position.setY(tu->position[1]);
  queries[num_queries].position.setZ(tu->position[2]);
  queries[num_queries].heading.setX(tu->heading[0]);
  queries[num_queries].heading.setY(tu->heading[1]);
  queries[num_queries].heading.setZ(tu->heading[2]);
  queries[num_queries].id=module_id;
  queries[num_queries].master=master;

  num_queries++;
}

/***** Determine whether collisions have occurred between the segments
       of an axis and a plane.
*****/
void DetermineResponse(void){
   Vector planeNormal(0,1,0);      /*normal to the plane*/
   Vector planePoint(0,-0.08,0);   /*point on the plane*/
   double planeSpring=20.0;        /*spring constant (Hooke's Law)*/
   Vector point;                   /*furthest point between a segment*/
                                   /*and the plane if collision occurs*/
   Vector distance;                /*distance from plane to point*/
   Vector line;                    /*direction of computed force*/
   double force=0;                 /*magnitude of force*/
```

```
    Vector localForce;              /*magnitude and direction of force*/
    Cmodule_type *comm_symbol;      /*holds info sent to plant model*/
    comm_symbol=new Cmodule_type[num_queries]; /*allocation of space*/

    for(int i=0; i<num_queries; i++){
       queries[i].force.set(0,0,0);          /*initialize force to zero*/
       comm_symbol[i].num_params=5;    /*datastructure has 5 parameters*/
       for(int j=2; j<5; j++){        /*send back the last 3 parameters*/
          comm_symbol[i].params[j].set=1;
          comm_symbol[i].params[j].value=0;
       }
    }

    for(i=0; i<num_queries; i++){                    /*for each segment*/
       if(find_collision(queries[i], &point, planeNormal, planePoint)){
          distance.projection(point, planeNormal);   /*shortest vector*/
          force=planeSpring*distance.length();    /*magnitude of force*/
          line.normalize(planeNormal);            /*direction of force*/
          localForce.scalarMult(force, line);/*magnitude and direction*/
          queries[i].force=localForce; /*apply this force to segment i*/
       }
    }

    for(i=0; i<num_queries; i++){        /*for each segment, send force*/
       comm_symbol[i].params[2].value=queries[i].force.getX();
       comm_symbol[i].params[3].value=queries[i].force.getY();
       comm_symbol[i].params[4].value=queries[i].force.getZ();
       CSSendData(queries[i].master, queries[i].id, &comm_symbol[i]);
    }
}

/***** finds if a collision occurs between a point and a plane
*****/
int find_collision(item_type query, Vector *point,
                   Vector planeNormal, Vector planePoint){
    double value;                       /*returns 1 if collision occurs*/
    Vector n1, n2, p1, p2, temp;        /*temporary vectors*/
    Vector endPoint;                    /*end point of segment*/

    endPoint.scalarMult(query.length, query.heading);      /*end point*/
    endPoint.add(query.position, endPoint);

    /***check to see if endPoint is below plane***/
    n1=planeNormal;                                  /*normalize vectors*/
    n1.normalize(n1);
    n2=query.heading;
    n2.normalize(n2);

    p1.projection(n1, n2);           /*use method of detection that is*/
    p2.sub(n1, p1);                           /*described in Chapter 7*/
    p2.normalize(p2);
    temp.scalarMult(-query.radius, p2);
```

```
    point->add(temp, endPoint);
    point->sub(*point, planePoint);
    value=n1.dotProduct(*point);

    if(value<=0)                          /*return 1 if collision occurs*/
       return(1);
    else
       return(0);
}

/***** Controls the loop of data exchange
*****/
void MainLoop(void){
  Cmodule_type two_modules[2];
  unsigned long module_id;
  int master;
  CTURTLE turtle;

  for(;;){       //infinite loop - until signal 'exit' comes
    CSBeginTransmission();
    num_queries=0;

    /*process data*/
    while(CSGetData(&master, &module_id, two_modules, &turtle))
       StoreQuery(master, module_id, two_modules, &turtle);

    DetermineResponse();

    if(CSEndTransmission()) break;  //End transmission
  }
}

/***** Main function
*****/
int main(int argc, char **argv){
  CSInitialize(&argc, &argv); //initialize the communication
  MainLoop();
  CTerminate();               //should be last function call
  return 1;
}
```

The functions `StoreQuery()`, `MainLoop()`, and `main()` take care of the communication process and are provided by the communication library `comm_lib.h`. The `StoreQuery()` function saves information that is sent from the plant model to the environment into the datastructure `queries[]`. Specifically, the radius of the cylinder is saved in `queries[].radius`, the length of the cylinder is saved in `queries[].length`, the position of the turtle is saved in `queries[].position`, and the heading vector of the turtle is saved in

`queries[].heading`. The functions `MainLoop()` and `main()` control the loop of data exchange (see [28]).

The function `DetermineResponse()` detects any collisions that occur between segments of an axis and a plane and computes the resulting forces. The variable `comm_symbol` of type `Cmodule_type` (which is defined in `comm_lib.h`) is used to transfer data from the environment to the plant model. The first `for-loop` in the function initializes `comm_symbol` so that the appropriate data will be returned to the plant model. The second nested `for-loop` detects whether a collision occurs between segments of an axis and a plane and computes the forces that result from any occurring collisions. The function `find_collision()` performs the calculations that specify whether a collision has occurred. The forces resulting from collisions are stored in `queries[].force`. The last `for-loop` assigns the computed force values to `comm_symbol`. Function `CSSendData()` transfers the data that is stored in `comm_symbol` to the plant model.