The modelling of growing natural forms

Huw Jones* and Jaap Kaandorp†

Lansdown Centre for Electronic Arts, Middlesex University, London, UK
Faculty of Mathematics, Computer Science, Physics & Astronomy, University of Amsterdam,
Amsterdam, The Netherlands

Abstract
The tutorial introduces a variety of ways of modelling branching biological structures. Methods for generating plant like images, such as Diffusion Limited Aggregation and Iterated Function Systems are briefly discussed. The main concentration is on two methods that can be used to model the growth of such structures. The powerful L-systems method gains explanatory power by use of parameters, context and environmental sensitivity, enabling natural features such as leaf drop, self-pruning and self-seeding. The method of Kaandorp for modelling marine sessile organisms such as sponges and corals, in which geometric features are “grown” directly, is compared for veracity with observed organisms in different physical environments.

1. Introduction
This tutorial is structured in three parts. Part one describes a variety of methods for producing images of plant like forms. This discusses the natures of various types of depiction, gives a historical overview of specialised studies and examples of Iterated Function Systems (IFS) and Diffusion Limited Aggregation (DLA). Part two concentrates on L-systems. Refinements to the basic method that enable modelling of growing structures are described and the section ends with two case studies of fungus and tree growth. References for parts 1 and 2 (by Jones) appear at the end of part 2. Finally, the modelling of growing marine sessile organisms using geometric transformation rules which relate to environmental factors is described in part three (by Kaandorp). This technique is shown to have close similarities with observations from nature.

1.1. Realism or not?
The style of depiction should depend on the potential use of the image. In flight and driving simulators, movement may mask inadequacies. Many systems use “billboarding” to represent trees as one or more intersecting scanned planar images rotated always to face the location of the observer (as by Visual Environments Inc. 5). Architectural line drawings may depict trees as simple polyhedral structures, to match the complexity of other features (fig. 1.1) (by the late John Lansdown 6). The rough cross-hatched texture of computer generated trees by Campa 7 looks natural in particular types of architectural illustration (fig. 1.2).

Poynter 8 developed a scheme for rapid production of images as interpretations of cartographic data sets (fig. 1.3). Terrain is depicted from height information. Individual fields are delineated and texture mapped according to their crops,

Figure 1.1: Polygonal trees in an architectural drawing

Figure 1.2: Campa’s sketchy trees in an architectural illustration
1.2. Some historical developments

Computer models of growing processes should be soundly based in biology. Classics by Thompson and Stephens and web sites are useful and readable. The Register of Ecological Models lists downloadable ecological modellers.

Honda gives a parametric description of tree skeletons. He distinguishes monopodial branching (in which one sub-branch continues in the same direction as the originating branch) from dichotomous branching. He influenced the work of Aono and Kunii, which preceded the popularisation of and is similar to the well known L-systems method, described in detail later. They model a variety of genuses of tree, and give data that enables the results to be duplicated. Smith describes the method of “graftals”, also an independently developed method related to L-systems. The name he chose connects the near fractal structure of trees to their graphic depiction. Trees are fractal to the extent that they display “statistical self similarity”, with branches and sub-branches having scaled down structural likeness to the tree as a whole. However, this is seen in nature to be limited to about seven levels of sub-copies, rather than the infinitesimal subdivision allowed in fractal theory. This observation can be useful in limiting tree models based on fractal methods. Estvanik gives detailed information on implementation of graftals.

Bloomenthal models tree branches as generalised cylinders whose axes follow spline curves. Control points for branches are created by a recursive algorithm. A high degree of realism is achieved by texture mapping of bark and leaves, and level-of-detail features are included for viewing at different distances.

Reeves and Blau illustrate collections of trees using particle systems to add leaves to an underlying branch structure. The structure is generated recursively, with separate algorithms applied to affect growth by features such as gravity, prevailing wind and sunlight. A typical forest image requires of the order of one million particles, which are sorted and rendered with colours taking into account sunlight and shade. The emphasis is on creating an effective image rather than directly modelling botanical data.

De Reffye et al. describe what is now known as the AMAP method (Atelier de Modélisation de l’Architecture des Plantes). Françon and Lienhardt describe its goal as: the definition of a rigorous and faithful model of plants and plant growth, for botanical and agronomical purposes. The method is based on botanical knowledge, and is validated through observations and measures on natural plants (p 27).

The method takes account of whether growth is continuous or not, how branches develop, whether branches have a tendency to grow towards the horizontal (plagiotropy) or vertical (orthotropy), and is based on tropical tree data collected by Hallé, Oldeman and Tomlinson.

Holton models a tree structure as a collection of unbroken strands running from the tree base (or root) to terminate at a leaf or branch. The thickness of a structural element depends on the numbers of strands passing through it. Gravity and nutrition supply are also taken into account.
Weber and Penn\textsuperscript{\textcircled{23}} use curved and tapering cones for the trunk, with branching structures defined recursively. They list 80 parameters used to control structure in their system for each of four different tree types. These govern features such as overall tree shape, size, levels of recursion, angles of branching and number of leaves per shoot. They also list the following features of a “good” tree modelling and rendering system, which they claim are achieved by their method:

- an appropriate level of resolution and quality;
- three dimensional modelling to enable realistic animation;
- accurate branching structure;
- capability of modelling a wide variety of tree types;
- random variations;
- time-dependent variability due to natural forces;
- concise and easily understandable modelling structure;
- models of different resolution to be viewed at different distances;
- stable and easy-to-use system.

Cellular automata, in which the contents of pre-defined cells change at regular clock ticks using functional rules dependent on the contents of neighbour cells, have been used in a number of plant studies. Arvo and Kirk\textsuperscript{\textcircled{24}} and Greene\textsuperscript{\textcircled{25}} use them to “train” climbing plants against existing structures. In particular, Greene uses strands, as Holton did later, to give varying thickness in roots negotiating their way around rocks. Colasanti and Hunt\textsuperscript{\textcircled{26}} create a 2D growing environment through cellular automata, in which the contents of cells change at regular clock ticks depending on the contents of neighbouring cells. The above ground cells are initially loaded with a parameter representing sunlight energy, which subsequently “seeps” down through cells, with cells in shade when growing structures cells above them absorb light. Below ground cells are loaded with soil nutrient levels, which are absorbed by growing roots, to be replenished in a simulation of natural processes. Branching circuits are defined to contain trees, shrubs, rock or bare soil with given levels of moisture. Fire affects these levels and spreads in a way controlled by local terrain slope and strength and direction of the wind.

Benes uses location information and a shadow buffer to assess the amount of light received by growing parts of an L-system structure\textsuperscript{\textcircled{27}}. This is applied to structures after the method of de Reffye et al.\textsuperscript{\textcircled{15}} to influence growth rates and to bend structures towards sources of light\textsuperscript{\textcircled{28}}.

The movement of plants affects the realism of a display, even in still images, where distortion due to wind affects plant geometry. Stösser et al.\textsuperscript{\textcircled{30}} show animations of weeping willows in the wind. Jones and Aitken\textsuperscript{\textcircled{31}} use a wind-tree model to animate simple stick and leaf representations of trees (fig. 1.5). Their model uses pseudo physics to allow elastic recoil and twisting of branches, fluttering of leaves, wind attenuation for segments in the lee of others and varying wind forces. Wu et al.\textsuperscript{\textcircled{32}} build their models of trees by user interaction to create outlines of and give three dimensional depth to selected branches of a photographic image, which is then animated. Ono\textsuperscript{\textcircled{33}} animates deformation and destruction of trees under severe wind forces. Although the method is simplified, there is no branch twisting, wind attenuation nor collision detection. Ono states, “visually satisfying results do \textit{not} depend on physical accuracy”. The results were shown to be “perceptually realistic” in the 1996 film \textit{Twister}.\textsuperscript{\textcircled{34}} The Tree Storm system\textsuperscript{\textcircled{35}} animates trees at varying levels of wind stress.

This is a selective choice of important or original work in the modelling and rendering of plant growth. Other papers worthy of note cannot be discussed in detail due to limitations of space. They include the work of Honda et al.\textsuperscript{\textcircled{36}}, the pioneering work of Kawaguchi\textsuperscript{\textcircled{37}}, Erickson\textsuperscript{\textcircled{38}} on the geometry of phyllotaxis, Gardner\textsuperscript{\textcircled{39}} on textured quadric surface representation of trees and later\textsuperscript{\textcircled{40}} fractal ellipsoids in a paper that concentrates mainly on representation of smoke, Viennet et al.\textsuperscript{\textcircled{41}} on ramifying patterns, Sakai\textsuperscript{\textcircled{42}}, Castéra and Morlier\textsuperscript{\textcircled{43}} on structural influences and L-systems to branching structures\textsuperscript{\textcircled{44}}, Fournier et al.\textsuperscript{\textcircled{45}}, Takenaka’s consideration of light sensitivity\textsuperscript{\textcircled{46}}, Farnsworth and van Gardingen’s study of the Sika spruce\textsuperscript{\textcircled{47}}, Deussen et al.\textsuperscript{\textcircled{48}} give an important review of the state of the art in plant modelling, discussing ways of achieving memory and time efficiency. Products, such as Tree Professional\textsuperscript{\textcircled{49}}, Virtual Forest\textsuperscript{\textcircled{50}}, PlantStudio\textsuperscript{\textcircled{51}} and xfrog\textsuperscript{\textcircled{52}} offer high quality tree modelling, with tree features input and controlled interactively. Before considering the two main themes of this tutorial in detail, we first look at two other methods used for depiction of plant like forms.

\subsection*{1.3. Iterated Function Systems}

Iterated Function Systems (IFS) are described here as they are commonly used to depict plant like forms. These are images without any descriptive power in terms of growth, so only a brief description is given. IFS were largely developed and publicised by Barnsley\textsuperscript{\textcircled{52}} and his collaborators as a technique for generating fractal forms and for image compression\textsuperscript{\textcircled{53}},\textsuperscript{\textcircled{54}}. There are two main methods for generating an image of an IFS.

An IFS is defined by a set of contractive transformations, \{T\textsubscript{1}, T\textsubscript{2}, \ldots, T\textsubscript{n}\}, often affine transformations that may be constructed by combinations of the translations, scalings and rotations familiar to computer graphics experts\textsuperscript{\textcircled{55}}. These result in simple linear equations. If all scale factors lie...
between -1 and 1, the transformation is contractive in that distinct points will be moved closer to each other under transformation. There are two methods for generating images or objects of an IFS. The first involves applying the transformations to a raster image, starting with an arbitrary neutral image \( I \).

Given arbitrary images \( I, J \)
Repeat
Copy \( I \) into \( J \) and clear \( I \)
Loop for \( k = 1 \) to \( n \)
Draw \( T_k(J) \) into \( I \)
End \( k \) loop
Until \( I \) and \( J \) are similar
Display \( I \)

The sequence of composited images stabilises after some iterations to the ‘attractor’ of the IFS, regardless of the form of the neutral starting image, just as the “steady state” solution of a differential equation can eliminate the memory of the “initial conditions”.

![Figure 1.6: A simple IFS example after four iterations](image1)

Figure 1.6 shows how a simple IFS can be related to the image or attractor, which has become relatively stable after four iterations from an initially all black image. The left part of fig. 1.6 shows an image region (outer square) and four affine transformations as interior rectangles. The transformed image is created by overlaying the scaled, translated and rotated versions of the original image that fill these four regions. Comparison with the IFS attractor identifies transformations corresponding to the lower trunk, lower left branch, lower right branch and upper part, each a contracted copy of the whole “tree”.

Generating an image from an IFS is easy, the inverse problem of finding the IFS that represents an image is not so simple. Although the process can be automated, its relative slowness due to complexity has been one of the major drawbacks of IFS as an image compression system. The ‘collage theorem’ is useful in identifying component transformations. If the image can be recreated by overlaying copies of itself, each of these copies represents a component transformation of the IFS. Figure 1.6 illustrates this, the location of sub-copies of the object itself relating to the positions of transformed rectangles.

The alternative image generation method is known as the ‘chaos game’.

Given arbitrary point \( P_0 \)
Loop for \( k = 1 \) to a large number
Select \( T_k \) at random from \( T_1, \ldots, T_n \)
Find point \( P_k = T_k(P_{k-1}) \)
Plot point \( P_k \)
End \( k \) loop

A Markov chain \(^{39}\) of points eventually fills the attractor that comprises the image, provided enough iterations are allowed. The image is more evenly filled if transformations are selected with probability proportional to their area contraction ratios, thus generating more points in large compared to small regions. This merely affects the speed of achieving a steady solution, in any case a deterministic object results from a random process. The fundamental theorem of IFS is that one and only one image is generated from this algorithm. By looping enough times, both algorithms produce identical images of the same IFS.

The illustration of fig. 1.6 is based in two dimensions, tree like objects occupying 3D space can be created by three dimensional transformations (fig. 1.7). Clearly, this method mimics appearance and has no information on growth characteristics. Local changes to an image or object are difficult to achieve, as the result of a change to one of the transformations will be distributed throughout the whole structure due to the fractal self-similar property of the IFS.

Colour as well as position may be transformed, and can be made dependent on the transformation selected, for example, brown for a ‘trunk’ transform, green for a ‘foliage’ transform. In ‘distributed IFS’, the probability of selection of transformations is dependent on the previous selection.

Several authors use non-affine transformations to extend modelling options. Gröller \(^{37}\) adds tapering and twisting functions, Frame and Angers \(^{38}\) use higher level polynomials, Jones and Campa \(^{39}\) use randomised functions and Jones and Mour \(^{40}\) use functions involving moduli (fig. 1.7). Branches are flattened and folded by a modulus function to mimic the features of a naturally occurring tree. 3D rendering of random points uses z-buffer and shadow buffer. Hart \(^{61}\) states that, ‘even the contractivity condition can be weakened to so called eventual contractivity’.

Figure 1.7: A 3D IFS using non-affine transformations

### 1.4. Diffusion Limited Aggregation

Diffusion Limited aggregation (DLA) generates branching structures using a stochastic algorithm. \(^{52, 63, 64}\). This represents structures that grow by accumulating particles, which “stick” to parts of the existing structure. Aharony \(^{40}\) gives electrodeposition, growth in aqueous solutions, dielectric breakdown and viscous fingering in porous media as

© The Eurographics Association 2000
Figure 1.8: Examples of diffusion limited aggregation with different walk and aggregation rules

Examples. These non-botanical features are mentioned here as physically growing branching structures. Guyon and Stanley give several illustrations of natural and synthetic examples.

A starting structure (often a single pixel) is defined as occupied. A particle is set off in a random walk from a randomly specified origin, typically at a set radial distance from the original particle. If the particle moves too far from start pixels, it is discarded and a new one created. If the particle lies adjacent to an occupied pixel, it is aggregated to the structure, and its pixel is labelled as occupied. The process continues until the structure reaches a given size, often defined by a maximum distance from the original particle (the diffusion limitation). The process produces branched clusters (fig. 1.8).

In the three top illustrations of fig. 1.8, the aggregation rule is four-connected, a pixel is only absorbed to the structure if it lies alongside an edge of an occupied pixel. The lower row uses an eight-connected aggregation rule, allowing edge and corner connections. The columns have different walk characteristics, the leftmost has a four-connected walk, with equal probabilities for up, down, left or right moves. The middle column uses an equal probability eight-connected walk, with diagonal moves also allowed. Noting that diagonal steps are longer by a factor of $\sqrt{2}$, the rightmost column “weights” the probability of a diagonal step by $1/\sqrt{2}$ to make the expected length (probability times distance) of a step in any direction the same. This attempts to compensate for grid dependency, which is visible in the general vertical/horizontal or diagonal tendency of growth directions in some images. It would be useful to test this compensation statistically rather than visually.

Rules can be varied in several ways. Objects may be grown outwards from inside the occupied cells. If the initial attractor is a row of cells with the start of the random walk at another parallel row, plant like growths from the “ground” attractor are achieved. Figure 1.9 shows this effect in the upper illustration, where lower growths appear stunted by overshadowing of more successful structures. The other images are similarly set up with aggregation inhibited until a particle has been in an “aggregating position” five (middle) or ten (lower) times, as though it is accumulating stickiness. Increasing the number of required visits gives greater bulk to the aggregation.

Batty and Longhurst model the growth of cities using a “dielectric breakdown model”, in which walk probabilities are affected by a field function which takes account of terrain slopes and natural barriers such as rivers, and accumulation is delayed until receptor sites have achieved a number of visits. Stanley shows how the field itself can be affected by the current state of the aggregate. Roberg and Abbess use DLA within a Manhattan grid road layout to model the accretive growth of traffic jams. Figure 1.10 shows a large traffic jam simulation created by a blockage at one intersection of a Manhattan grid road layout, which shows oddly plant like growth features.

Figure 1.9: Line attracted DLAs. From top to bottom, particles are accumulated after one, five and ten visits

1.5. Summary

This concludes a brief review of various methods of modelling the appearance and growth characteristics of growing objects. The main methods of L-systems and the modelling of marine sessile organisms are explored in greater depth in the two following parts.

2. L-Systems

Lindenmayer devised L-systems as a set of mathematical construction rules for describing plant forms as character strings. They may also be used to construct regular fractal forms. Levy describes how two of Lindenmayer’s students, Hogeweg and Hesper, generated computed images of L-systems which Lindenmayer considered ‘charming but inconsequential’. Smith, stimulated by a Lindenmayer talk at a conference in Holland, subsequently developed his related method of graftals. Lindenmayer was convinced of the value of visual interpretation of L-systems through his collaboration with Prusinkiewicz from the mid 1980s.
grammars but L-systems apply productions "growth". The method is similar to Chomsky’s transformational applied to all characters of the existing string, enabling strings. At a sequence of finite clock ticks, productions are rules for converting single characters to other characters or strings. At each clock tick, productions are applied to all characters of the existing string, enabling “growth”. The method is similar to Chomsky’s transformational grammars but L-systems apply productions simultaneously to all characters at each clock tick, rather than serially.

An L-systems string can be illustrated by relating the alphabet to the turtle graphics commands of the Logo language. The position of a “turtle” that moves around the drawing surface is controlled by string characters. The position of a turtle is given by its location and the direction it faces, \((x, y, \theta)\) and moves are made relative to this position. The command “F” for “move forward” is enabled by changing the turtle definition to \((x + d \cos(\theta)), y + d \sin(\theta), \theta)\), where \(d\) is a set distance for the move — \(d\) may be associated with \(F\) as a parameter, \(F(d)\). Logo also includes the commands “pen down” and “pen up” to enable drawing of lines or position change without drawing, some systems use “F” for the “pen up” version of “F”. Direction change is created by “+” for turn left, “–” for turn right (many authors use these in the opposite sense). These may be defined to have a set angle, or may be parameterised, so \(+(\alpha)\) and \(–(\theta)\) change the turtle definition to \((x, y, \theta + \alpha)\) and \((x, y, \theta - \alpha)\) respectively. Characters “[“ and “]” mean “start a branch” and “end a branch”. When the string is interpreted as a drawing, the character “[“ does not affect the turtle, but pushes its current state onto a stack data structure. Drawing continues until the corresponding ”]” is reached, the turtle is moved in “pen up” mode to the location popped from the stack, then it is returned to “pen down” mode to continue drawing from the location where the branch emanated. The stack ensures proper nesting of branches, so sub-branching is achieved to the required level.

Suppose the alphabet consists of characters:
- \(F\): forward drawing step,
- \(+\): left turn of 45˚,
- \(–\): right turn of 45˚,
- \(<\): left turn of 30˚,
- \(>\): initiate a branch,
- \([\): end a branch.

Let the axiom be “F” and establish the single production

\[ F \rightarrow F [ \times F ] [ < F ] [ F - F ] F, \]

which means that character “F” is changed to the string to the right of the arrow. By convention, when a character has no defined production, it is left unchanged, it is subject to the production * → *.
Specialist features can be added to the basic system to overcome the problems identified above. These include stochastic drawing and productions, context sensitivity, query modules, parametric L-systems and three dimensional L-systems.

2.1. Stochastic drawing and productions

Randomness is used to give natural irregularity to physical location and structures, to geometry and topology. The former is achieved by subjecting the turtle state to random change, affecting length and angle of drawing elements, giving the kind of variability created by natural causes in real plants.

This is not straightforward in implementation. Suppose we want to draw the ‘tree’ of fig. 2.1 with increasing levels of detail by reducing the drawing length by one third at each stage, but also to include some stochastic angle variation. At stages 1 and 2, the string to be drawn is

1. \[ F [+F] [+F] [−F] F \]
2. \[ F [+F] [+F] [−F] F [+F] [−F] F [−F] F \]

The bracket before the second branching section is italicised, it is at string position 6 at stage 1, at position 32 in stage 2. When the branch angle is implemented, it will be given some random deviation, when its location in the list is varied, is not straightforward. The random value allocated can be stored as a parameter pointed at from the character when it is first generated in the list. However, the starting location (turtle position vector (x, y, z), you face the direction 'left' direction fig. 2.4). When you sit in an aircraft, your vectors, Orientation is given by three mutually orthogonal unit vectors, \( \mathbf{r} = (x, y, z) \) in a 3D Cartesian space. Position vector \( \mathbf{r} = (x, y, z) \) represents displacements \( v_x \), \( v_y \), \( v_z \) respectively in the \( x, y \) and \( z \) directions. A unit vector represents a displacement of overall unit length, which

\[ \mathbf{u} \]

meaning that with probability 0.05, \( F \) is transformed in the same way as previously, but with probability 0.05, \( F \) is replaced by a null segment, which terminates drawing of the particular branch and subsequent segments, as discussed later. Figure 2.2 shows seven examples of this effect, allied to random angle deviations for each \( F \) stage of -2˚, 0˚ and 2˚ applied with equal probability. The 'plants' now have irregularity both in location and form.

This was the first run of the programme, readers may believe there are only six examples shown. The small speck to the left is the first example, killed off by random chance, the L-systems equivalent of a late frost. This effect has never subsequently occurred in any run of the programme. Figure 2.3 shows instances from a similar L-system, the left/right turn directions are selected at random and there is a slight tendency for branches to turn towards the upward vertical, simulating the orthotropic growth of some plants (those that tend to the horizontal are plagiotropic). The effect is to produce plant like images that differ from each other but could be from the same genus. These examples show more natural features than the deterministic methods of fig. 2.2.

2.2. L-systems in three dimensions

Simple extensions of the alphabet and its interpretation allow generation of L-systems in 3D, allowing a greater degree of realism. For a 3D turtle, location is given by position vector \( \mathbf{r} = (x, y, z) \) in a 3D Cartesian space. Orientation is given by three mutually orthogonal unit vectors, \( \mathbf{h} \) defining heading, \( \mathbf{u} \) for the “up” direction and \( \mathbf{l} \) for the “left” direction fig. 2.4). When you sit in an aircraft, your location is \( (x, y, z) \), you face the direction \( \mathbf{h} \) with the top of your head pointing towards \( \mathbf{u} \) and the craft’s left wing pointing towards \( \mathbf{l} \). (For those not familiar with vectors, a 3D Cartesian vector \( \mathbf{v} = (v_x, v_y, v_z) \) represents displacements \( v_x \), \( v_y \), \( v_z \) respectively in the \( x, y \) and \( z \) directions. A unit vector represents a displacement of overall unit length, which

\[ \mathbf{u} \]
applied to all features of an object to be reoriented. Using them computed, so that only one standard operation needs to be example, to rotate of a vector the vector result of this, we get properly defined. These standard operations are easily coordinate system to represent position \((x, y, z)\), so the \(z\)-axis is angle about the \(z\)-axis. We are already using a Cartesian graphics matrix methods \(^5\). The use of rotation), with rotations implemented by standard computer rotation), plus ‘–’ for a negative rotation, with rotations implemented by standard computer graphics matrix methods \(^5\). The use of \(\mathbf{h}, \mathbf{u}\) and \(\mathbf{l}\) to define the turtle orientation has considerable redundancy, as it could be defined by the three “Euler” angles, representing the rotation operations required to rearrange the coordinate axes in the turtle orientation (Euler was a prolific 18th Century Swiss mathematician). However, by holding this extra information, redirection of the turtle is simplified, so this is one of the many examples in computer graphics when memory is traded for speed and simplicity of operation.

Suppose matrix \(\mathbf{M}\) represents a composite rotation aligning a general unit vector \(\mathbf{v}\) with the \(z\)-axis, \(\mathbf{l}\), is the inverse of this (realigning the \(z\)-direction with \(\mathbf{v}\)) and \(\mathbf{R}\), is a rotation of given angle about the \(z\)-axis. We are already using a Cartesian coordinate system to represent position \((x, y, z)\), so the \(z\)-axis is properly defined. These standard operations are easily implemented in a 3D computer graphics system, and can be concatenated, or combined to make a single operation \(^5\). For example, to rotate of a vector \(\mathbf{w}\) by a general angle about a general vector direction \(\mathbf{v}\), the sequence

- use \(\mathbf{M}\) to align \(\mathbf{v}\) with the \(z\)-axis,
- use \(\mathbf{R}\), to rotate by the required angle about the \(z\)-axis,
- use \(\mathbf{l}\), to realign the \(z\)-axis with the direction of \(\mathbf{v}\)

is applied to \(\mathbf{w}\). Using the standard methods of matrix operation, the order of application is important. Thus, applying \(\mathbf{M}\), \(\mathbf{R}\), \(\mathbf{w}\) is done by the matrix multiple \(\mathbf{M}, \mathbf{R}, \mathbf{w}\). Applying \(\mathbf{R}\) to the vector result of this, we get \(\mathbf{R}, \mathbf{M}, \mathbf{w}\). Finally, the application of \(\mathbf{l}\), gives the final direction as vector \(\mathbf{l}, \mathbf{R}, \mathbf{M}, \mathbf{w}\).

Combinations of matrices, such as \(\mathbf{I}, \mathbf{R}, \mathbf{M}\), can be pre-computed, so that only one standard operation needs to be applied to all features of an object to be reoriented. Using them as building blocks, the required direction changes of roll, pitch and yaw can be performed as follows.

**Roll** rotates by a given angle about the heading \(\mathbf{h}\), resetting left vector \(\mathbf{l}\) and up vector \(\mathbf{u}\) to \(\mathbf{I} = \mathbf{I}, \mathbf{R}, \mathbf{M}, \mathbf{I}\) and \(\mathbf{u}' = \mathbf{h} \times \mathbf{I}\). Figure 2.5 illustrates the four stages starting with the original location, subsequently applying rotations \(\mathbf{M}, \mathbf{R}, \mathbf{I}\), in turn. Note that in this case it is slightly less efficient to apply the matrix rotation method for \(\mathbf{u}'\) as even after pre-computation of \(\mathbf{I}, \mathbf{R}, \mathbf{M}\), used for \(\mathbf{I}\) this takes 9 multiplications, whereas the cross product \(\mathbf{u}' = \mathbf{h} \times \mathbf{I}\) takes 6 multiplications. Cross multiplication also ensures that \(\mathbf{u}'\) and \(\mathbf{I}\) are orthogonal, avoiding possible drift due to rounding error after repeated calculations. However, we note that the levels of branching in trees seldom exceeds seven, so the rounding errors are unlikely to accumulate significantly.

The operations of pitch and yaw are achieved in equivalent ways.

**Pitch** (tilting the aircraft nose downward from a normal flight position) rotates by the given angle about the left vector \(\mathbf{l}\), resetting \(\mathbf{h}\) and \(\mathbf{u}\) to \(\mathbf{h}' = \mathbf{I}, \mathbf{R}, \mathbf{M}, \mathbf{h}\) and \(\mathbf{u}' = \mathbf{h}' \times \mathbf{l}\).

**Yaw** (turning to the left) rotates by the given angle about the up vector \(\mathbf{u}\), resetting \(\mathbf{h}\) and \(\mathbf{l}\) to \(\mathbf{h}' = \mathbf{I}, \mathbf{R}, \mathbf{M}, \mathbf{h}\) and \(\mathbf{l}' = \mathbf{u} \times \mathbf{h}'\). Repeated applications of such sequences could cause rounding errors in lengths of the ‘unit’ vectors, as well as the angle drift mentioned above. However, this is unlikely to cause significant problems if the level of branching is no greater than \(7\).

Combinations of roll, pitch and yaw can orient tree branches in any desired direction. The symbols representing these can be included in the L-systems alphabet, and appropriately used in productions to mimic the characteristics of existing trees, for example the numbers of branches emanating at each branching point and the angles at which these branches occur.

**Figure 2.4:** The components of a 3D turtle

**Figure 2.5:** The roll operation in four stages

The Eurographics Association 2000
Figure 2.6 shows an example of a 3D tree generated by Briggs 
using conic frusta to represent forward ‘drawing’ steps. This 
has growth simulation features to be discussed later.

2.3. Context sensitivity

Further modelling complexity is introduced by context sensitive 
L-systems. Productions are applied dependent on the string 
characters that occur to the left or right of the subject. The ‘context’ here is restricted to the character string, effects of the 
larger geometric context in which the structure is situated are 
discussed later.

Representation of a context sensitive production is in the
form

L < C > R → Word.

The character ‘C’ is changed to ‘Word’ if ‘L’ appears to its left 
in a string and ‘R’ appears to its right. Prusinkiewicz and 
Lindenmayer \(^{75}\) call this a 2L-system. Systems which use only 
one sided context sensitive productions, of form 

L < C → Word

or

C > R → Word,

are called 1L-systems, and the concept can be widened further 
to IL-systems, in which the left and right contexts can be strings 
of defined length rather than just single characters.

Care must be taken in defining the left and right contexts in 
the case of bracketed L-systems. If a closing bracket 
immediately precedes the subject character or an opening 


the character ‘F’ has left context ‘B’ and right context ‘I’. ‘C’, 
also, has left context B. This technique is used by 
Prusinkiewicz and Lindenmayer to propagate signals along the 
structure, from root to tips (acropetally) using the left context, 
from tip to roots (basipetally) using the right context. As a

simple example, suppose we have an alphabet of two 
characters, • and –. With one axiom

and productions

• > – → –,
– → •.

The dash character is passed along the row of symbols from 
right to left in the sequence,

• • • • – • •
• • • • – •
• • • – • •
• • – • •

and so on. A dash is always changed to a dot, a dot is 
changed to a dash whenever it has a dash to its right. Some 
examples given below combine context sensitivity with other 
features.

2.4. Parametric L-systems

Since Lindenmayer proposed the association of numerical 
parameters with alphabetic characters \(^{85}\), the method has 
been highly developed. In most computer languages, it is 
relatively easy to create structures that define each character 
with one or more parameters. At the simplest level, these can 
represent obvious characteristics of the structural segment, 
such as its age (by incrementing time steps) or its length 
(perhaps scaled depending on the level of recursion at which 
it is generated). This is a possible method to increase realism. 
Lansdown \(^{2}\) urges caution, discussing a study \(^{86}\) which 
suggests internodes (distances between successive branching 
sections) are ‘more or less of fixed length’. Lansdown also 
cites a rule of McMahon, that the widths of a branch is 
proportional to its length to the power 3/2 \(^{87}\). Many such 
empirical studies are useful in relating models to real 
structures.

Productions now take two forms. Some affect parameters 
without changing the structure of the character string and 
thus the object structure. Structural change is made 
dependent on the values of certain parameters, for example a
leaf may be discarded if its age reaches a threshold, a branch 
may be formed if the accumulation of nutrient in its parent 
branch is adequate to trigger new growth.

Several examples are discussed in more detail later, just 
one simple illustrative case is shown here. Suppose a leaf 
with character L is to disappear after 20 developmental 
stages, and that it is created with initial ‘age’ 0, represented 
as L(0). A composite production to generate this effect is

L(t) : t = 20 → null,
L(t) : t < 20 → L(t-1).

The condition appears after the colon and before the arrow. 
Age can be used to affect size for growing features and to 
affect colour for decaying features, leaves give an example in 
both cases.

2.5. Environmental sensitivity

Now we look at the final bullet point of section 3, the 
separation of topological structure from geometric 
information. The simple L-system method is: “create a string,
then draw it’. There is a systematic driving set of rules that creates the object regardless of its location. This is clearly unrealistic in terms of growing plants, where environmental considerations can have considerable influence from the major effects of climate to the localised effects of the plant’s geometry itself. For increased explanatory power, the topological structure has to be made responsive to the location of plant elements.

Use of “query modules” was explained by Prusinkiewicz et al. 77 in an elegant paper describing synthetic topiary.

... 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. (p 352).

The locations of structural elements are made available via query modules as parameters at the time productions are applied, and thus may be used to affect the production as do other parameters. Similar results can be achieved by making the location values direct parameters of the structural element itself. Note that it is not necessary to ‘draw’ each developmental stage, recursion can continue for the required number of stages before drawing is performed, but locations are evaluated at each stage.

In their 1994 paper, Prusinkiewicz et al. 77 use the query module to simulate topiary. One or more tree growths are initiated inside the reference structure of a geometric model, either a simple shape such as a cube (fig 2.7) or cylinder or an isosurface model of a more complex shape such as a dinosaur. The latter method creates a particular function of position $\phi(x, y, z)$, defining the surface as the implicit solution of the equation

$$\phi(x, y, z) = \text{constant}.$$

The apical or growing tip of each branch is queried at each stage of development to identify its position. This is then tested against the reference structure to determine if it lies inside or outside. In the case of an issosurface model this is simply done by evaluating the field function at the point and comparing it with the field constant. When the tip pierces the structure, its branch is pruned and grows no more. The structure grows until it reaches or just passes the object’s surface, gradually filling the structure in a simulation of topiary. By making the growth sensitive to an environmental feature, the plant structure is affected by location.

In a subsequent development, Mëch and Prusinkiewicz 78 use the query module to affect growth dependent on plants’ own structures and the structures of other objects in the neighbourhood. Again, a query module is used at growing parts of branches. This time the location is tested against shadows cast by the tree itself and other neighbouring trees that are themselves growing. Rays are cast in general sun directions through the growing object and its environment to estimate part or full shadow volumes. The method is similar to that used independently by Aitken 31, 89 to identify branches in the lee of the wind from others in animating wind blown trees. If a branch section is not in total shadow, a signal is propagated back down the branch using context sensitive productions. Any branch section that does not receive such light energy for a given number of time periods is deemed to have died out, and is removed or culled from the structure — a counting parameter is used to determine the time since energy was received. The effect is to thin out overshadowed parts of the structure by self-pruning while allowing those that receive full sunlight to grow unhindered. This is a natural feature observed in forests, where interior trees develop into ‘canopy’ growths with the remains of stunted branches seen on otherwise bare trunks, full foliage only achieved where the canopy reaches clear sunlight. At the edge of the forest, individual trees have asymmetric growths, with leaves proliferating on the unshaded side.

2.6. Continuous time methods: Differential L-systems

Prusinkiewicz et al. 76 defined differential L-systems (dL-systems) to cope with continuous and discrete developmental aspects of growth. Leaves and new branches lengthen and thicken in continuous time, the development of a bud into a flower or branch is a discrete action. In discussing parametric L-systems above, it was noted that some productions change parameters such as lengths, others affect structure. Three points are made in favour of this new method.

- In principle, the time interval can be made arbitrarily small, but once chosen it is part of the model and cannot easily be changed. From the viewpoint of computer animation, it is preferable to specify the time interval as an easy to control parameter that is not implicit in the underlying model.
Fungi grow through their propagating medium in the form of moulds as branching networks of fine threads. These are 3 dimensional but can be visualised as overlapping curves. A whole network is called a mycelium with individual threads called hyphae (fig. 2.8). A simple model of a hypha is of a fine tube filled with a liquid (the cytoplasm), comprising all the chemicals and structures needed to sustain and

2.7. Modelling of Nutrient Distribution in a fungus

The remainder of this section describes two case studies in modelling of developmental growth by simulating features of natural development. Tunbridge and Jones 97 used the method of parametric L-systems to model a specific form of fungal growth, Aspergillus Nidulans, which develops as a set of filaments or growing tubes.

Their “bio-mechanistic” method involves direct modelling of the biological processes underlying the growth of the plant, showing that methods already used to model flowering plants, sponges and algae can be extended and adapted to the modelling of fungal growth.

Prosser and Trinci 98 contains a comprehensive review of early work on mathematical models describing the growth of filamentous fungi or moulds, but there have been few computer graphics simulations of such growths. Two examples are based on empirical models. The fungus Mucor hiemalis is modelled by Hutchinson et al 97 using a frequency distribution approach to model branching frequencies, angles and lengths. A computer model realistically determines colony morphology, but images produced use straight lines to represent curved filaments. Quinn and Fujimoto 98 use a similar method for the fungus Erysiphe polygonii, giving more realistic images through randomly determined angle changes in filaments. Liddell and Hanson 99 use an L-systems type environment with structures subject to free form deformations, and demonstrate the modelling of a dandelion and animations of growth. Kaandorp 84 also uses geometric construction rules, his work is described in considerable detail later in this tutorial.

The remainder of this section describes two case studies in modelling of developmental growth by simulating features of natural development. Tunbridge and Jones 97 used the method of parametric L-systems to model a specific form of fungal growth, Aspergillus Nidulans, which develops as a set of filaments or growing tubes.

Their “bio-mechanistic” method involves direct modelling of the biological processes underlying the growth of the plant, showing that methods already used to model flowering plants, sponges and algae can be extended and adapted to the modelling of fungal growth.

Prosser and Trinci 98 contains a comprehensive review of early work on mathematical models describing the growth of filamentous fungi or moulds, but there have been few computer graphics simulations of such growths. Two examples are based on empirical models. The fungus Mucor hiemalis is modelled by Hutchinson et al 97 using a frequency distribution approach to model branching frequencies, angles and lengths. A computer model realistically determines colony morphology, but images produced use straight lines to represent curved filaments. Quinn and Fujimoto 98 use a similar method for the fungus Erysiphe polygonii, giving more realistic images through randomly determined angle changes in filaments. Liddell and Hanson 99 use an L-systems type environment with structures subject to free form deformations, and demonstrate the modelling of a dandelion and animations of growth. Kaandorp 84 also uses geometric construction rules, his work is described in considerable detail later in this tutorial.

The remainder of this section describes two case studies in modelling of developmental growth by simulating features of natural development. Tunbridge and Jones 97 used the method of parametric L-systems to model a specific form of fungal growth, Aspergillus Nidulans, which develops as a set of filaments or growing tubes.

Their “bio-mechanistic” method involves direct modelling of the biological processes underlying the growth of the plant, showing that methods already used to model flowering plants, sponges and algae can be extended and adapted to the modelling of fungal growth.

Prosser and Trinci 98 contains a comprehensive review of early work on mathematical models describing the growth of filamentous fungi or moulds, but there have been few computer graphics simulations of such growths. Two examples are based on empirical models. The fungus Mucor hiemalis is modelled by Hutchinson et al 97 using a frequency distribution approach to model branching frequencies, angles and lengths. A computer model realistically determines colony morphology, but images produced use straight lines to represent curved filaments. Quinn and Fujimoto 98 use a similar method for the fungus Erysiphe polygonii, giving more realistic images through randomly determined angle changes in filaments. Liddell and Hanson 99 use an L-systems type environment with structures subject to free form deformations, and demonstrate the modelling of a dandelion and animations of growth. Kaandorp 84 also uses geometric construction rules, his work is described in considerable detail later in this tutorial.

The remainder of this section describes two case studies in modelling of developmental growth by simulating features of natural development. Tunbridge and Jones 97 used the method of parametric L-systems to model a specific form of fungal growth, Aspergillus Nidulans, which develops as a set of filaments or growing tubes.

Their “bio-mechanistic” method involves direct modelling of the biological processes underlying the growth of the plant, showing that methods already used to model flowering plants, sponges and algae can be extended and adapted to the modelling of fungal growth.

Prosser and Trinci 98 contains a comprehensive review of early work on mathematical models describing the growth of filamentous fungi or moulds, but there have been few computer graphics simulations of such growths. Two examples are based on empirical models. The fungus Mucor hiemalis is modelled by Hutchinson et al 97 using a frequency distribution approach to model branching frequencies, angles and lengths. A computer model realistically determines colony morphology, but images produced use straight lines to represent curved filaments. Quinn and Fujimoto 98 use a similar method for the fungus Erysiphe polygonii, giving more realistic images through randomly determined angle changes in filaments. Liddell and Hanson 99 use an L-systems type environment with structures subject to free form deformations, and demonstrate the modelling of a dandelion and animations of growth. Kaandorp 84 also uses geometric construction rules, his work is described in considerable detail later in this tutorial.
develop the fungus. Such microscopic structures include nuclei, important ‘control centres’ for growth processes. Hyphae extend by growing at their tips. Branch hyphae are usually originated some way behind the tip. Hyphal tubes are divided, in some types of fungus, by walls called septa. A septum may form a complete barrier or may allow some cytoplasm to flow through a septal pore.

A new mycelium typically starts with the germination of a spore to form a single hypha or germ-tube. By a process of elongation and branching a colony is formed, generally lying roughly within a circle. Radial colony extension rate is elongation and branching a colony is formed, generally lying within a circle. Radial colony extension rate is approximately constant, whilst the total length of the mycelium increases exponentially.

The comprehensive mathematical model of fungal growth developed by Prosser and Trinci fits well into the formalism of context sensitive parametric L-systems. It is based on the underlying biological processes that support growth. This mechanistic model is summarised below and is used to develop the L-systems implementation of this study.

2.7.1. Cytoplasm formation and distribution

The cytoplasm contains vesicles, cytoplasmic structures containing nutrients necessary for extension of the hyphal or axil tip, which can be considered as tiny spheres suspended in the cytoplasm. They are produced throughout the hypha and flow towards the hyphal tip. It is believed that vesicles fuse with the hyphal tip when they reach it, their contents being used to fuel the extension of the tip. There are alternative theories, but the method used here still holds as a valid model of the distribution of ‘nutrients’ within the cytoplasm.

Trinci describes the formation of new septa (cross walls). The apical compartment, that portion of hypha that lies between the tip and the next septum, contains four nuclei. As the tip extends, the volume of cytoplasm contained in this compartment increases. When the volume of cytoplasm per nucleus in an apical compartment reaches a critical value, the four nuclei divide to form eight nuclei. When this division is complete, a new septum is formed across the apical compartment, separating the nuclei into two clusters of four. The septum gradually grows to its maximum size, giving increased resistance to the flow of vesicles.

2.7.2. Branching

When the number of vesicles in part of a hypha between two septa reaches a critical concentration, vesicles fuse with part of a side-wall of the hypha causing it to swell and eventually branch at that point. Septa have a constraining effect on the flow of vesicles, so this usually occurs in the part of a hypha just behind a septum. Flow of vesicles through the hypha before the branch is divided so that some flow into the branch, the remainder flowing in the original direction.

2.7.3. Spore germination

The growth of a colony in this model starts from a spore. The spore produces vesicles at a constant rate until their concentration reaches a critical level, when the spore germinates by producing a hyphal tip. In the model, only one hyphal tip is produced by an individual spore, but this can be modified to suit different circumstances.

2.7.4. The mathematical model

This has two main elements, to control production of hypha and vesicles and to control the hyphal extension through absorption of vesicles. The former considers the hypha as consisting of tips and segments. Segments have no biological analogue, being used purely for modelling purposes except in the case of apical segments, which are collectively equivalent to an apical compartment. General hyphal segments have constant length \( L_s \) and produce vesicles at a constant rate \( R_p \).

Tips increase in length as they absorb vesicles, being initially of negligible length. When the tip length reaches that of a hyphal segment, a new hyphal segment is created behind it, reducing the tip to negligible length. Vesicles, produced at a constant rate by each hyphal segment, flow at a constant rate towards the tip. The presence of a septum inhibits flow. The number of vesicles absorbed by a tip in one standard time interval is functionally dependent on the number of vesicles in the tip, as well as the preset saturation constant for absorption and maximum rate of absorption. Elongation of the tip in the time interval is proportional to the number of vesicles absorbed.

Values of relevant constants are taken from observations by Prosser and Trinci as specified in \(^9\), which contains detailed descriptions of the L-systems productions, only qualitative descriptions are given below to describe the nature of the processes involved.

2.7.5. L-systems implementation

The L-systems representation of Aspergillus Nidulans has sixteen context sensitive parametric productions (five of which affect structural change, the other eleven affect parameter values) on an alphabet containing nine characters, listed below.

- P: ungerminated spore,
- G: germinated spore,
- T: tip,
- A: apical segment,
- S: normal segment,
- M: segment with a septum
- B: segment from which a branch is produced,
- [: branch start,
- ]: branch end.

Figure 2.9 shows a small part of a mycelium represented as

\[
\]

All characters except brackets ‘[’ and ‘]’ have associated parameters, from one for \( P \) up to six for \( T \).

P(0) is the axiom, an ungerminated spore with no vesicles. A parametric production increases vesicle level at a constant rate up to a threshold, when \( P \) is changed to a germinated spore \( G \) and a growing tip with appropriate parameters. \( P \) takes no further role, \( G \) continues to generate and export vesicles.

None of the parameter changing productions is stochastic, the model could be improved by allowing some random variation in vesicle production. Most characters have parameters representing their current vesicle content and the number of vesicles to be exported to the right context (\( T \) does not have the latter). The character list is scanned from the
Figure 2.9: A small part of a mycelium represented by L-systems characters

left, imports from the left context being added to the current content by context sensitive productions, calculating the amount to be exported as a fraction of the current content. This simulates the motion of vesicles towards the growing tips according to Prosser and Trinci’s model 96. For a branch base B, the export is divided proportionately into two parts, one to pass into the right context septum (M), the other to continue into the branch (S or A). It should be clear that context sensitivity is an important aspect. Structure affecting productions are (ignoring parameters for simplicity):

- \( P \rightarrow GT \) an ungerminated spore germinates and grows a tip when its vesicle content achieves a threshold;
- \( T \rightarrow AT \) a tip grows, leaving an apical segment behind it when it achieves a certain size;
- \( A \rightarrow M \) an apical segment becomes a septum on a signal that its right context has achieved a certain size;
- \( A \rightarrow S \) an apical segment becomes a normal segment when its right context has changed to a septum;
- \( S \rightarrow B[T] \) a normal segment becomes a branch segment with a growing tip when its parameter value achieves a threshold.

The model was checked against the experimental observations of Prosser and Trinci 96, and gave a high level of agreement with the number of growing tips and the overall length of hypha generated over a range of growing times. both differed by less than 10% from observed values up to 16 hours simulated growth 95.

2.7.6. The drawing stage

The method is of the limited form in which the character string is created independently of the geometric location of the structure. This is unrealistic, in that the growth probably develops to seek nutrient in, for example, a Petri dish. An enhancement of the method would be to ‘grow’ the mycelium in a culture of nutrient, possibly represented as cellular automata with nutrient content being diminished by the presence of hypha in a cell. The mycelium could then be given a tendency to grow towards levels of higher nutrient, and vesicle production could be made environmentally sensitive. This should achieve the relatively circular nature of such structures in reality. In the absence of this, a number of pragmatic random drawing rules are adopted to give a reasonable visual interpretation.

A spore, P or G, is drawn as a small circle. Structural characters representing hyphal segments are drawn as forward turtle lines. A, S, M and B, have fixed length, T has length proportional to its vesicle content, which is held as one of its parameters. Stochastic drawing elements include:

- random branching direction (right or left);
- Normally distributed branching angle;
- randomly deviated turtle direction for segment drawing, with a bias towards the radially outward direction.

The last condition is a pragmatic attempt to simulate the outward search for nutrient, that is only partly successful as can be seen in fig. 2.10. However, the model is successful in simulating many of the features of a natural form of growth using parametric and context sensitive L-systems.

2.8. A tree growth simulation

The work developed by Tunbridge is extended by Briggs, 84, 106, 107 to the modelling of growing trees using context and environmentally sensitive L-systems. This study does not attempt to generate a particular genus of tree, but creates a general tree-like growth, with many naturalistic features. Further work is needed to fine-tune the model to specific tree types.

A fungus grows by outward distribution of cytoplasm. A growing tree distributes nutrient absorbed from the soil by its roots. This depends on soil water saturation and chemical content. It also generates and distributes energy in the form of sugars and starches formed by photosynthesis. This is dependent on the time span and intensity of sunlight received at leaves and, to a lesser extent, the trunk and branches. Growth is powered by nutrients and energy. Leaves, when they age, discolour and fall. New trees are propagated from seeds that fall from the original plant. All these features are included in the model, in which trunk and branch segments...
grow thicker by absorbing nutrients and energy distributed about the structure. (fig 2.11).

Figure 2.11: Conceptual illustration of nutrient and energy flow in the L-system model

2.8.1. The L-systems specification

Branching and direction changes are enabled by the usual symbols. Other characters (listed below) represent root R, stem S, apical (or growing) bud A, leaf L (fig 2.11), G (ground), p (query module) and % (prune). Their parameters and associated productions are given below, all structural changes are triggered at a parameter threshold $84, 106$. The axiom is essentially

$$R S \{ L \} A,$$

a root and one stem segment with an apical bud above them and a leaf emanating from the top of the stem segment. The full specification is

$$R(1) S(1, 0, 0, 0) \{ \rightarrow L(1, 0, 0)p(0, 0, 0) \} A(0).$$

Each leaf has a query module $p$ associated with it. This is loaded with the $(x, y, z)$ location of the leaf’s base when the system develops, but is initiated with a zero value.

- **Root** $R$(export nutrient)
  
  The root could be the base of a second L-system representing root structure, but this has not been implemented. The root exports nutrients to its right context, which must be a stem segment $S$. The thickness of a stem segment is related to Holton’s strands $22$, its first parameter being the number of “strands”. The number of strands (like a bundle of strings) is proportional to the cross sectional area, so $S$ has diameter proportional to the square root of its number of strands, upper radius equal to the base radius of the right context. At branches, the passage of nutrients is divided in proportion to the number of strands in the main stem and branch section.

- **Stem** $S$(strand, nutrient, export nutrient, export energy)
  
  A stem segment imports nutrient from its left context (root or stem), imports energy from its right context (when it is a stem) and any subsequent branch section (leaf or stem), exports nutrients to its right context (apical bud or stem) and to any existing branch segment (leaf or stem) and exports energy to its left context (stem or root). It has no structure changing productions; once created, it remains in place. The simplified production

$$S(s, n, e_n, e_e) \rightarrow S(s + n, importNutr*N + importEnergy*E, (1 - N)*importNutr + D*importEnergy, (1 - D - E)*importEnergy)$$

represents twelve context sensitive productions which evaluate the effects of nutrient imported from the left context (importNutr) and energy imported from the right context (importEnergy) for all combinations of left and right contexts of character S. Parameters $n$, $e_n$, and $e_e$ represent the nutrient store, nutrient for export and energy for export.

Proportions $N$ of imported nutrient and $E$ of imported energy are absorbed, enabling thickening. Parameters $E$, $D$ (a fraction of imported energy converted into nutrient for development) and $N$ affect the general tree shape. Each stem section is depicted as a frustum of a cone, base radius proportional to the square root of its number of strands, upper radius equal to the base radius of the right context. At branches, the passage of nutrients is divided in proportion to the number of strands in the main stem and branch section.

- **Apical Bud** $A$(nutrient)
  
  An apical bud is the growing tip of a branch or trunk. It is depicted as a cone whose height is proportional to its nutrient content, with base width of one strand. It accumulates imported nutrients from the left context until a threshold, $g$, required for growth is achieved.

$$S(s, n, e_n, e_e) < A(v): (v \leq g) \rightarrow A(v + e_n).$$

When $g$ is passed, the apical bud is converts to the structure of fig. 2.12, creating a new branch and two leaves. It is depicted as a cone whose height is proportional to its nutrient content, with base width of one strand. When $v > g$, the production is essentially (ignoring parameters)

$$A \rightarrow \{ S \{ L \} A \} S \{ L \} A,$$

adding one forward stem segment with branching leaf and apical bud, with an equivalent side branch structure.

$$S(s, n, e_n, e_e) < A(v): (v > g) \rightarrow A(v + e_n).$$

A stem segment imports nutrient from its left context (root or stem), imports energy from its right context (when it is a stem) and any subsequent branch section (leaf or stem), exports nutrients to its right context (apical bud or stem) and to any existing branch segment (leaf or stem) and exports energy to its left context (stem or root). It has no structure changing productions; once created, it remains in place. The simplified production

$$S(s, n, e_n, e_e) \rightarrow S(s + n, importNutr*N + importEnergy*E, (1 - N)*importNutr + D*importEnergy, (1 - D - E)*importEnergy)$$

represents twelve context sensitive productions which evaluate the effects of nutrient imported from the left context (importNutr) and energy imported from the right context (importEnergy) for all combinations of left and right contexts of character S. Parameters $n$, $e_n$, and $e_e$ represent the nutrient store, nutrient for export and energy for export.

Proportions $N$ of imported nutrient and $E$ of imported energy are absorbed, enabling thickening. Parameters $E$, $D$ (a fraction of imported energy converted into nutrient for development) and $N$ affect the general tree shape. Each stem section is depicted as a frustum of a cone, base radius proportional to the square root of its number of strands, upper radius equal to the base radius of the right context. At branches, the passage of nutrients is divided in proportion to the number of strands in the main stem and branch section.

$$S(s, n, e_n, e_e) < A(v): (v \leq g) \rightarrow A(v + e_n).$$

When $g$ is passed, the apical bud is converts to the structure of fig. 2.12, creating a new branch and two leaves. It is depicted as a cone whose height is proportional to its nutrient content, with base width of one strand. When $v > g$, the production is essentially (ignoring parameters)

$$A \rightarrow \{ S \{ L \} A \} S \{ L \} A,$$

adding one forward stem segment with branching leaf and apical bud, with an equivalent side branch structure.

$$S(s, n, e_n, e_e) < A(v): (v > g) \rightarrow (90) \rightarrow (90) \rightarrow (24) S(1, 0, 0, 0)$$

$$S(1, 0, 0, 0) \rightarrow (90) L(0.5, 0, 0)p(x, y, z) \rightarrow (90) S(1, 0, 0, 0) \rightarrow (90) L(0.5, 0, 0)p(x, y, z) \rightarrow (90) L(0.5, 0, 0)p(x, y, z) \rightarrow (90) S(1, 0, 0, 0),$$

$A(v + e_n, g).$
Is the full rule for this production, showing angle change and initial parameter values allocated in the 3D model, showing that new stems have one strand.

This production can be changed to match characteristics of specific tree types to model particular species, it is the “engine” that drives the tree’s topological structure.

- **Leaf** \( L(\text{scale}, \text{age}, \text{export energy}) \)

The leaf is the important photosynthetic element, importing nutrient from and exporting energy to its left context. Exported energy is the product of imported nutrient and a sun factor \( H \), which can be dependent on time, leaf age and on the location given in the leaf’s query module for shading effects. In practice, the energy produced diminishes with age of a leaf, and apical buds and other features of growing plants have a relatively minor photosynthetic effect. These features are not implemented, but the data structure allows for such developments in future implementations. The leaf is aged by a set amount \( D \) each stage. When the age of the leaf reaches 70\% of its optimum, it falls to the ground. This is implemented by replacing string element \( L \) by \( GL \), where \( G \) is a “ground” character, reducing the \( y \)-coordinate of the turtle location to zero for drawing of the next leaf element. When the leaf reaches twice its optimum age (by this stage, it is lying on the ground and coloured black), it is pruned from the string. The character ‘\%’ indicates that the character \( G \) to which it is applied and all subsequent characters up to the end of the current branch level (which can only be a leaf structure in this case) are eliminated. The leaf could be scaled dependent on a size parameter \( s \) (variation of \( s \) is not shown below) and its colour varies with age. The current implementation ranges through light to dark greens, browns, then red and black after falling to the ground. The following four productions create these effects.

\[
\begin{align*}
S(\ldots, \ldots, e_n, \ldots) &< L(s, a, n): (a \leq 0.7 - D) \rightarrow L(s, a+D, e_n*H); \\
L(s, a, n): (0.7 - D < a \leq 0.7) &\rightarrow GL(s, a+D, 0), \\
L(s, a, n): (0.7 < a < 2) &\rightarrow L(s, a+D, 0), \\
G > L(s, a, n): (a \geq 2) &\rightarrow \%
\end{align*}
\]

A basic leaf is pre-defined as a simple closed shaded kite. Size change for growing leaves has not been implemented, but would not be difficult. With symbol \( F(d) \) depicting a forward drawing edge of length \( d \), the leaf boundary is

\[ \{-(40) F(0.25) +(60) F(0.47) +(320) F(0.47) +(60) F(0.25)\} . \]

### 2.8.2. Examples of modelled trees

The OpenGL 1.0\* ancillary library is used to depict trunk sections as interpenetrating truncated cones with base radius and upper radii related to the strand parameter as described above.

Figure 2.13 shows two views of a tree that has grown for 100 cycles, with the constant settings \( N = 0.02, E = 0.01, D = 0.01, H = 1, W = 1 \) (these are defined within section 2.8.1).

Branching angles are subject to stochastic variation to give naturally irregular appearance. No clash detection has been implemented, so some branches may be seen to pass through each other on close scrutiny. It is worth reiterating that this feasibility study does not attempt to model a particular type of tree and that structural, as opposed to visual, effect is of major importance.

Different effects are produced by changing constants. With nutrient flow parameter \( N \) increased from 0.02 to 0.05, more nutrient is absorbed within stem segments, which grow more thickly. Less nutrient is available for leaf development, so the photosynthetic effect is diminished, producing a structure with thicker trunk and less foliage (fig. 2.14), which has parameters as for fig. 2.13 except for the change of \( N \) and the number of growth stages set to 120.

\( E \), representing energy absorption within stem sections, is increased from 0.01 to 0.04 in fig. 2.15, giving increased thickening of stems compared to fig. 2.13, this occurring more evenly from the leaves down in comparison with the ‘root up’ effect of fig. 2.14. In fig. 2.15, the number of stages is 120 and other parameters except \( E \) are as for fig. 2.13. This includes drawing parameters, such as lengths of cylinder
sections and random angle variations, so it is clear that N and E have a considerable influence on tree shape.

By associating a query module \(^7\) with each leaf, the effect of having part of the structure in shade can be simulated. In fig. 2.16, the sun energy factor H for leaves to the left of the tree trunk (where \(x < -1\)) is diminished to one tenth of the used for leaves to the right. The structural effect is clearly visible, with the left side being less well developed with thinner branches. Real tree branches in shade may develop to be long and thin in a “search” for light. Figure 2.16 does not show this characteristic, but it does show how growth is affected by environmental factors. This could be fine tuned to make length as well as thickness dependent on energy content. Leaf colours change with age and those older than 70% of the optimum age detach from the tree using the ‘ground’ symbol G. Real plants re-seed through the germination of fallen seeds. Flower, fruit and seed characters have been implemented in an incomplete study, but to simulate the effect of plant redevelopment, a randomly selected small proportion of the fallen leaves have been allowed to behave as seeds, generating new plant growths. Several leaves close to the trunk in fig. 2.17 have re-seeded. Figures 2.16 and 2.17 are developed over 90 stages, with parameters as for fig 2.13.

2.9. Summary

These case studies demonstrates the potential of parametric L-systems for modelling nutrient and energy flows in plant growth. There is potential for further development of the model. It would be relatively easily to include stochastic variation of growing parameters and use of observed day by day rainfall and sunshine data to simulate changing nutrient uptake from the soil and photosynthetic energy production. Other environmental data could be used to affect productions, adjusting formulae to take account of seasonal
Acknowledgments

Huw Jones is grateful to Aristic Lindenmayer and Przemyslaw Prusinkiewicz for their inspirational development of the L-systems method. Also, staff and students at Middlesex University have contributed in various ways to this work, particularly the late John Lansdown, who contributed through discussion and encouragement. Others who have worked with the author, and whose work or images are directly used, include Aurelio Campa, Stephen Poynter, Jon Thum, Michael Stuart, Matthew Aitken, Magnus Moor, Penna Roberg, Chris Abbess, Paul Briggs, Andrew Tunbridge. David Arnold and Russell Taylor of the University of East Anglia are thanked for access to their research references. Many who are not named contributed in less direct fashions, but their activity is nonetheless appreciated by the author.

References

34  Industrial Light and Magic. Twister, Jan de Bont (director), produced by Amblin Entertainment, distributed by Warner Brothers, 1996.


