# Modelling the Growth and Shape of Trees

John Coffey, Cheshire, UK.

2025

## 1 Introduction

Trees are the largest and amongst the most beautiful plants on our planet. Artists have drawn them for centuries by careful observation or creative abstraction, endeavouring to capture their essential forms, whether as isolated trees, in a copse or as a forest. This article is my our account of computer algorithms to model the growth and morphology of different species of trees. The subject has a long and varied history, being of interest to botanists, commercial tree growers and ecologists. CGI trees are used in computer games and in the cinema industry as part of a virtual world. My perspective is as a painter and 3D artist.

Broadly trees are distinguished from other plants by being

- flowering plants, producing fruit and seeds each season,
- being large, supported by the tough but fairly flexible fibrous woody cores of trunk and branches,
- living for a long time – some trees are 1000 years old,
- being especially useful to humans for food, timber, shelter and shade.

Trees fall into one of two broad categories: deciduous, which shed there leaves each autumn, and evergreen, which do not have a leaf-shedding season. A second distinction, made by woodworkers, is between hard woods and soft woods. Biologically this is a distinction based on microstructure, but it is broadly true that most hard woods are harder to work than most soft woods.

We all know that trees have a trunk rooted in the soil, large branches at some distance up the trunk, each of which subdivides one or more times until the distal structure is a cluster of twigs bearing leaves which capture sunlight and carry out photosynthesis. Seen from a distance a tree in summer looks like an irregular green ball. In art it can be coloured as a ball along with its shadow. Many children readily draw a shape which is recognisable as a tree. There will be regional differences; some will draw conifers like cartoon zig-zag Christmas trees, others will draw globular green oaks on a brown trunk or apple trees spotted with bright red balls as apples, and yet others may draw palm trees, such is the diversity of tree species, each adapted to its environment.

§2 gives a brief account of tree botany since the test of any model is how closely is depicts real trees. §3 reviews mathematical models using fractals, which was perhaps the earliest approach to tree modelling, developed in the 1960s and 70s. Closely related to fractals are L-systems, and these are outlined in §4. §5 examines biology-based models of tree growth over their annual cycles. Development of these was started in the late 1970s mainly by biologists and used to model ecology, optimise forestry, control woodland fires, and allied commercial and environmental applications. Since about 1990 computer power has developed so greatly that photo-realistic trees, foliage and other plants can be generated and rendered in computer games and cinematic CGI. CGI has driven tree modelling since about 2000, and has learned much from the plant physiology models developed by botanist, so-called 'functional-structural' tree models. §6 comments on several computer applications for 3D modelling of trees amongst these CGI developments, now widely used by games designers and special effects artists. I do not include a section specifically on the characteristics of common trees because there are many excellent accounts in books and on the internet.

## 2 Biology of tree growth and shape

This section reviews the basics of tree growth year by year. There are many good accounts of tree physiology on the internet[1], but here we only need a few essentials.

### Tree anatomy

The living parts of the tree are its roots, leaves, flowers and inner bark. The inner bark consists of two concentric layers of tubular cells, the phloem towards the outside and the xylem towards the inside. Phloem cells transport sugars created in the leaves down to the roots and out to the growing stems and branches, while xylem cells transport water and minerals from the soil upwards. Together they form the 'vascular cambium'. Both phloem and the more plentiful xylem cells are elongated fibres and both are short lived. When phloem cells die, they are pushed outwards to form part of the outer bark. When xylem cells start to die, they are pushed inwards by the more vigorous cells and form another cylindrical layer in the woody body of the tree. Xylem cells which are still alive but not functioning effectively form the soft sapwood, while dead xylem cells form the central hardwood. Hardwood cells are 'glued' together with lignin to form the strong but slightly flexible internal skeleton of the tree.

Each spring and summer a ring of new sapwood is added. The early faster growing cells are paler in colour than the later, more slowly growing ones. In temperate climates the cambium slows to zero growth during the winter, marking as a dark stain the end of each annular growth ring as seen in the felled tree in Figure 1. There are more xylem cells than phloem so almost all radial growth is in the sapwood and heartwood. The cork layer below the outer bark does grow slowly each year through addition of phloem cells, but the bark is simultaneously shed.

[1] See for instance 'Tree biology', Module 2 by the Tree Council, a UK charity. Also 'Tree Growth Characteristics' by Franklin and Mercker, University of Tennessee.

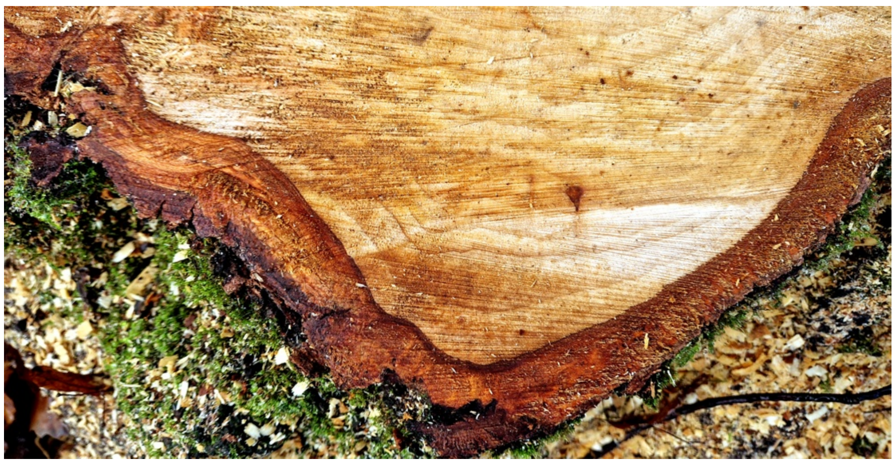


Figure 1: A sawn tree trunk showing bark and thin dark cambium over central sap and heart wood. Growth rings are just discernible.

**Growth**

Trees grow in a yearly cycle. In spring each bud on a twig will shoot forth a leaf and the twig will grow in length to separate the new leaves so they can best capture sunlight. This is 'apical' or primary growth. In addition twigs, stems, branches and truck will grow slightly in cross-section to increase their so that the structure can bear the extra weight and the fluctuating force of wind on the leaves. This is 'radial' or secondary growth.

Apical growth is concentrated close to discrete positions along a stem called 'nodes', illustrated in Figure 2, left. The node is where a leaf will form from a leaf bud. When any

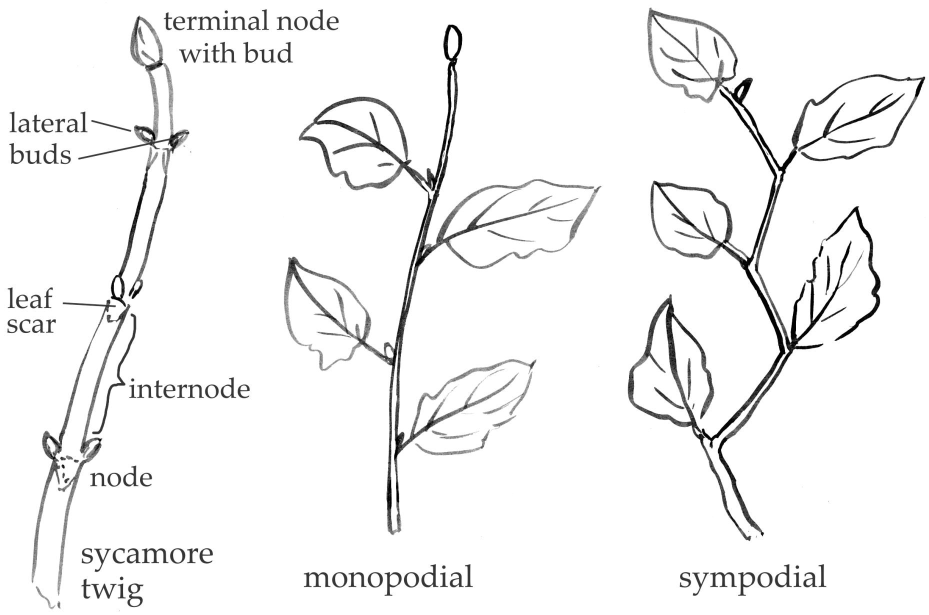


Figure 2: Growth from nodes. Left: twig with paired lateral buds. Centre: monopodial growth from terminal bud. Right: sympodial from lateral buds.

leaf along the twig dies in autumn, it leaves a small scar, and immediately above the scar is a ‘lateral bud’ which can sprout next year into a new branch stem or else lie dormant for years until needed.

Trees vary in the way nodes and hence leaves and daughter twigs are distributed along a stem. Some produce single leaves and lateral buds alternately to left then right, or in a spiral around and along the twig, while others produce leaves in pairs to left and right of each node as in the sycamore in Figure 2 left. The pattern of buds and leaves spiralling around the stem is called phyllotaxis. Furthermore, shoots can grow either straight or in a zig-zag fashion depending on what happens at the terminal bud. As Figure 2 shows, in straight ‘monopodial’ growth the terminal bud continues as the main growing point, but in zig-zag ‘sympodial’ growth the terminal bud produces a leaf or flower and stops growing there, requiring further growth to come from the nearest lateral bud. The pattern of nodes remains in each twig as it grows over the years into a branch, so the pattern of branches resembles the pattern of leaves along a stem.

The most important node is the one at the end of the twig, the terminal bud. In young trees one terminal bud will grow vertically upwards and so is called the ‘leader’. The stem immediately behind the terminal bud can grow faster than any other part of the twig, and the terminal bud of the leader can grow fastest of all. This has a strong influence on the shape of the final tree, as described below. Trees with and without a strong central leader are photographed in Figure 3.

The ratio of growth rate of the terminal bud to rate at lateral buds is called the ‘apical dominance ratio’. It is controlled by hormones produced at the terminal bud. Hormones particularly from the leader may inhibit growth of lateral shoots throughout the young tree, as Figure 4 illustrates. A high ratio means that the leader grows strongly straight upwards

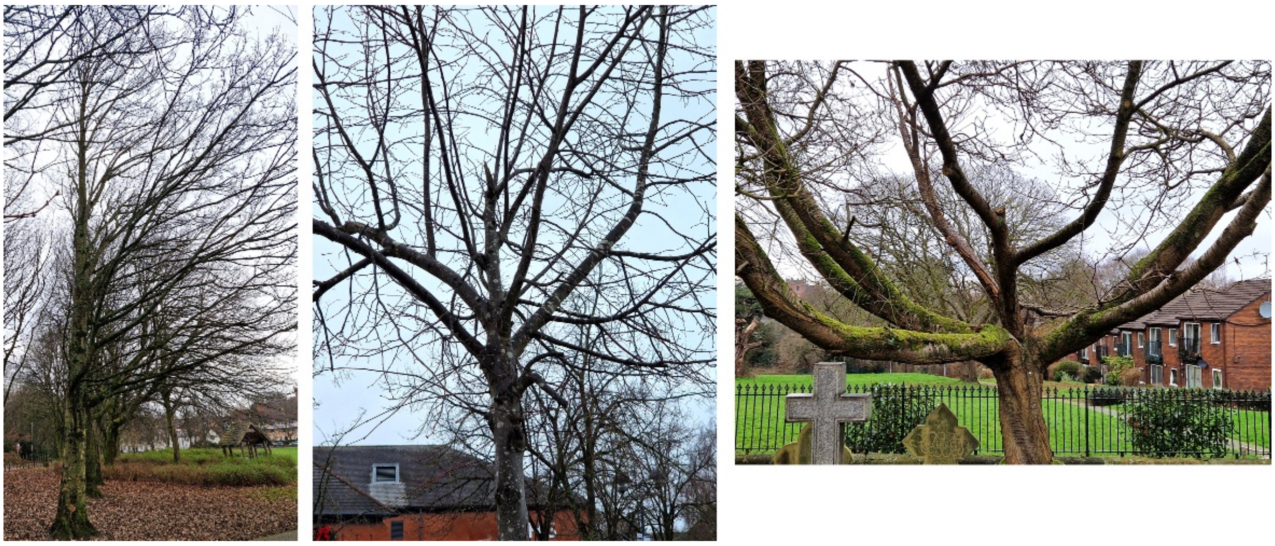


Figure 3: Left: tall tree in copse which has kept its leader. Centre: younger tree with snapped off leader, growing side branches. Right: old cherry in churchyard whose leader was lost long ago.

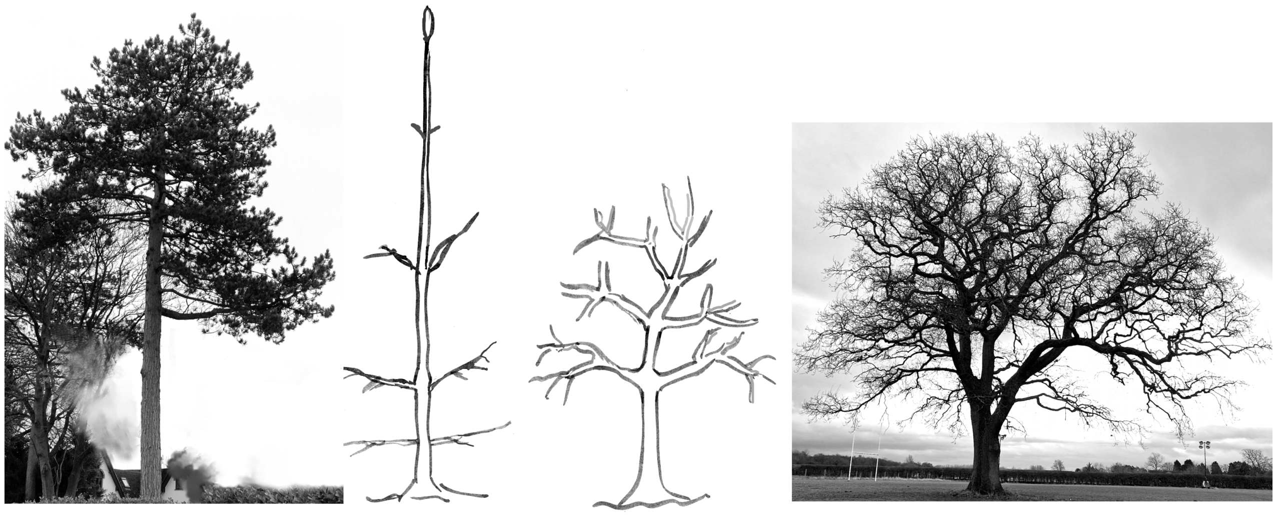


Figure 4: Effect of apical dominance ratio on tree growth. Left pair: pine with high ratio. Right pair: oak with ratio close to 1, showing curved and twisted branches.

with little growth of side branches. What side branches there are tend to grow horizontally. The tree overall grows tall and straight with no forking, as with pines and other conifer species. High apical dominance ratio therefore produces a conical tree shape. On the other hand, if the apical dominance ratio is low, near 1 as it is in many deciduous species, the terminal bud of each twig is no more significant than the lateral buds, so side shoots readily develop. The tree develops a forked trunk, forked branches and a wide spreading crown. This well describes an oak tree.

**Tree and root shape**

We see from the above that in several ways the small scale growth pattern in the leaves on twigs carries through to the morphology of the final mature tree. Six distinct crown shapes have been distinguished in trees growing in open land: round, oblong, oval, vase, pyramidal and weeping, all shown in Figure 5. Where trees are growing together in competition for light, however, the trunks are generally long and straight with few side branches. The high crowns of trees in a forest are collectively called the 'canopy'.

Branches are thinner but more plentiful towards the top of the tree. Leonardo da Vinci is credited with being the first to record the rule of thumb that when the trunk or a branch divides, the aggregate cross-sectional area remains essentially constant. For instance, if the trunk as it grows divides into two equal branches, the sectional area of each branch will be half that of the trunk, and hence its diameter about 70% of the trunk's. Three equal branches would each have diameter 58% and four branches 50% of the trunk's. Evidence from finite element modelling suggests that this optimises the tree's flexibility to strength ratio and so maximises resistance to wind damage.

In many species branches grow in an upwards curve. This is probably partly a consequence of phototropism in which sunlight inhibits growth, causing the upper, lit side of

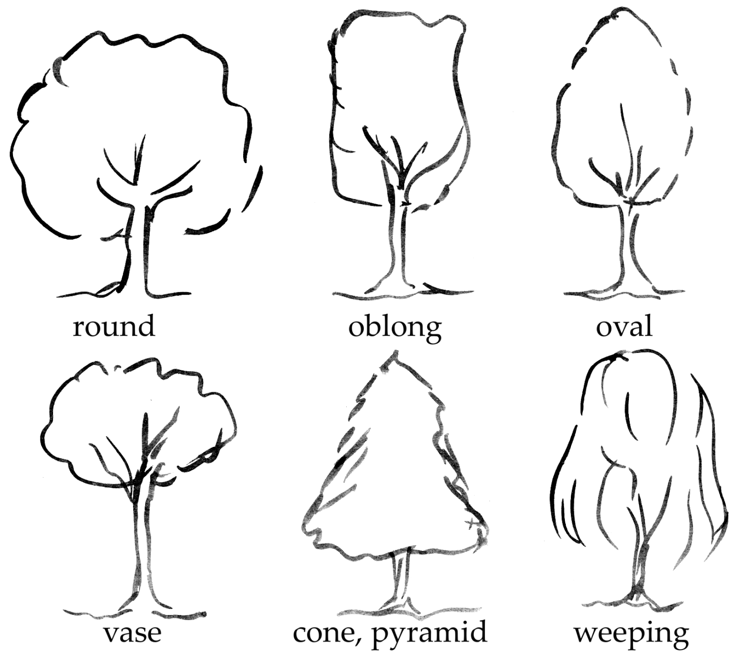


Figure 5: Six tree shapes proposed by Dwight McCurdy in 'How to Choose Your Tree', 1972. Oak is round, ash usually oval, holly, yew and many conifers are conical. Willow and birch are weeping.

each stem to grow more slowing than its underside. The other major influence is the bending stress distribution along the branch. Strong upwards curvature at the end of branches is a characteristic of ash and horse chestnut trees.

The root system also grows each year, with apical growth at nodes similar to twigs and branches. Typically almost all roots are in the top 2 feet (60 cm) of the soil surface, but cover an area around the tree which extends well beyond the drip line of the outermost leaves. The overall proportions of a tree on open ground – crown, trunk and roots – have been likened to a brandy glass standing on a sandwich plate, such is the extent of the root network. From an artistic point of view roots can be interesting and attractive features, since they are often clearly seen at the banks of streams and on steep ground where the lower soil has fallen away. Young trees grow with a tap root, but in most species that gives way to the wide but shallow network. A tap root can stabilise a tree against wind, so species which do retain a substantial tap root like oak are more likely to snap off even large branches in a gale rather than blow over, whilst flat rooted trees like beech and birch are more likely to blow over completely.

The shape of mature trees is also affected by the damage they suffers because of strong winds, and by the quality of the soil. Trees naturally shed a large fraction of their twigs, particularly those in shadier regions, in order to stop over crowding. This is called 'self-pruning' . When the wind snaps off twigs, dormant nodes further down the stem (towards the trunk) become active and may grow new twigs. These can be at a large angle to the original twig, almost a right angle, and this can lead over the years to the branch (as the twig has become) having a distinct elbow. If the wind has been strong enough to blow the whole tree over, some species have the ability to make roots from the branches which were previously horizontal but now stick vertically downwards into the soil. This is an example of vegetative reproduction.

In towns and public parks the shape of trees is much influenced by deliberate pruning. All lower branches will be removed to allow traffic to pass or let light into buildings. The crown may also be reduced by tree surgeons to stop encroachment onto buildings, or to limit the public liability risk of falling branches. However, many mature urban trees are subject to tree preservation orders. The conflict between the legal requirement to retain the tree and yet limit its growth have led to some ugly pruning of trees into tormented sticks.

**Energy use and damage repair**

Trees are said to become 'stressed' when the sunlight, water or soil conditions are not good. This can be understood in terms of the energy needs of the tree. Energy comes from captured sunlight through photosynthesis in the leaves.

$$\textit{Sunlight} + \textit{water} + \textit{carbon dioxide} \rightarrow \textit{glucose} + \textit{oxygen}.$$

The oxygen is released to the atmosphere and the glucose not required by the leaves themselves is transported by the phloem cells around the tree and into the roots. Each branch creates enough carbohydrate to maintain itself, with the excess being converted to starch and stored in the xylem of that branch, or transported to the truck and roots for storage. This starch is the energy reserves for the two most active phases in the yearly cycle: spring when new leaves form, and autumn when they are shed.

Within cells the chemical potential energy in glucose is converted back to energy to drive cell growth by the complementary process of respiration:

$$\textit{glucose} + \textit{oxygen} \rightarrow \textit{energy} + \textit{carbon dioxide} + \textit{water}.$$

The carbon dioxide is released back into the atmosphere and the energy used for apical, radial and root growth, the repair of damage, and the synthesis of chemicals which protect the leaves and bark from feeding animals, birds and caterpillars, and from insect infestation.

If any branch, or the tree as a whole, cannot produce enough stored starch, it may not produce enough leaves when spring comes, and so go into year on year decline. Similarly, if the soil becomes impacted and the roots cannot gain oxygen to respire, starch will not be reconverted into glucose for release as energy. Under such stress the tree may protect itself by preparing to shed even large branches which cannot produce enough leaves. This type of self-pruning is known as 'compartmentalisation', meaning that the deficient region of the tree in turned into a sealed off compartment; the tree seals off the phloem and xylem and activates chemical defences against disease. The sealed off branch gradually becomes weakened at its joint with the truck and may eventually snap off in the wind. Sealing minimises the extent and vulnerability of the wound. In open ground the naturally shed branches will mainly be towards the centre of the crown, resulting in a tree which has lost the top of its truck and has widely spaced sideways-growing branches. For the same reason branches near the ground may be shed, or never develop in the first place, if adjacent buildings or trees put them in constant shade.

**Some structural details**

I have not mentioned the shape of leaves because a painter would not paint them in detail unless such close-up were required. Conifers have needles or small scales for leaves. Oak, beech and many other deciduous trees have single leaves, some of iconic shape. Compound leaves come in two forms, pinnate and palmate as illustrated in Figure 6. Ash has pinnate leaves in which there are several pairs of leaflets either side of the central leaf vein. Horse chestnuts have palmate leaves which fan out from a central point. There is much on tree identification by leaf shape on the internet[2].

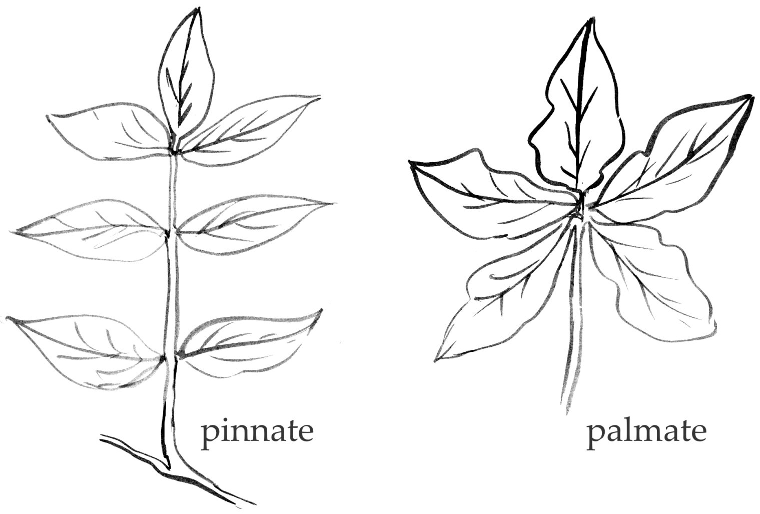


Figure 6: Two types of compound leaves. Ash are pinnate, horse chestnut palmate.

The pattern of fissures in the bark is characteristic of the species, and often enough for us to recognise it as beech, oak, or birch. Another characteristic of some barks are 'lenticels' – small lens-shaped opening in the bark to allow gas exchange. They are a marked feature of pear, cherry and birch trees (Figure 7).

If a large branch has snapped or been sawn off, the tree will repair the wound, leading to a bulging ring of scar wood around the break, as shown in Figure 7 right. Trees in towns have these rings all over their trunks.

One important structural detail is the joint between branch and trunk, or minor branch and main branch. These regions are reinforced by cross-crossed overlapping layers of xylem. On the top side of the joint the reinforcement will bulge into a ridge, possibly with cracks, and on the underside thicken into a raised collar around the trunk. If the branch has grown too upright and close to the trunk, the ridge may not be strongly formed and the branch therefore is inadequately secured and prone to breaking off. This latent split is seen in Figure 8, left, and both ridge and collar are illustrated in Figure 8, centre.

In many trees the phloem and xylem have a twisting, spiral shape up the tree which can also appear in the bark, as in Figure 8 right. This is pronounced in sweet chestnuts

[2] See for example the UK Woodland Trust website. Also *www.forestryengland.uk*

and can be seen in the splits in trunks of dead trees when the bark has fallen away. There is controversy as to why this occurs. One conjecture is that the spiral grain has increased strength and wind resistance over straight grain. Another is that a spiral is more effective in distributing water and nutriments around the tree, especially if it is growing on non-uniform ground, in a prevailing wind, or other directionally dependent local condition. Personally I have wondered whether it is a result of phototropism – the tendency of stems to grow towards the light – as the sun moves in the same direction across the sky day after day. Twisting also occurs in branches of oak and some other deciduous trees, partly on account of the low apical dominance ratio.

I close this section by reminding the reader of the amazing colour changes of leaves in autumn, caused by decreased chlorophyll and increased glucose storage which allows coloured pigments (related to carrots) to show through.

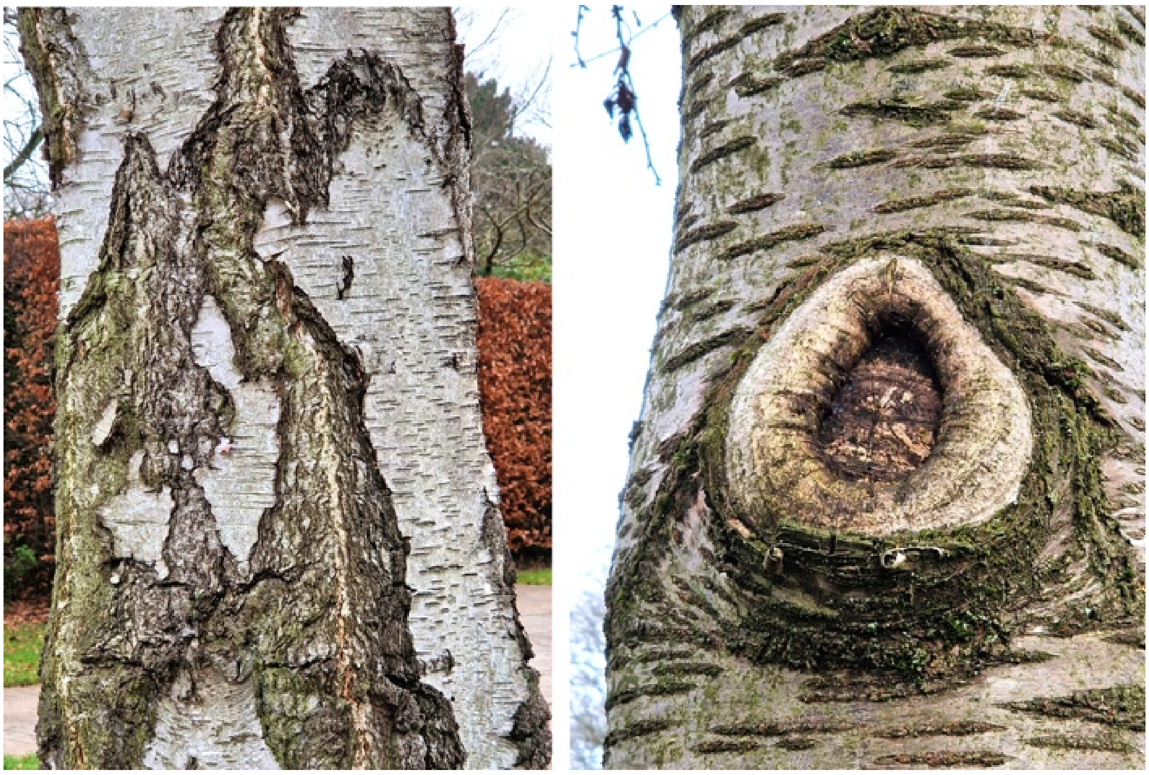


Figure 7: Left: horizontal lenticels and bark fissures in a mature birch tree. Right: lenticels and a wound ring in a cherry.

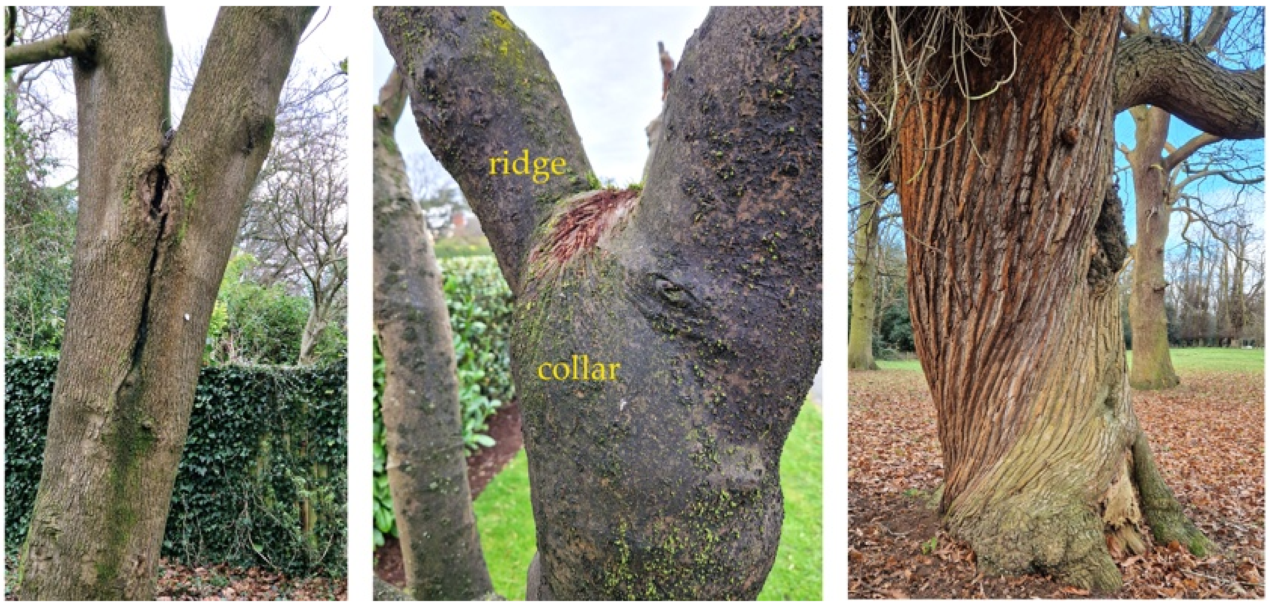


Figure 8: Left: weak branch joint in sycamore. Centre: detail of joint ridge with bark splitting and collar in laburnum. Right: a twisted trunk.

## 3 Fractal trees

Fractals are geometric forms which have several length scales and which appear the same or very similar on all length scales. That is to say, they look either exactly the same or, for random fractals, statistically the same, whether they are looked at by the naked eye, through a microscope or through a telescope. The classic fractals first investigated in the 1960s by mathematicians such as Benoit Mandelbrot were illustrations of the Julia set, solutions of cubic equations using Newton's iterative method, or abstract forms such as Sierpinski's triangular gasket. Fractals are a subset of chaos theory and dynamical systems. They are created by recursive algorithms in which a given structural unit is inserted in place of each element of itself an indefinite number of times. Mandelbrot was attracted by the controlled randomness in natural forms such as coastlines. He studied self-similar curves of infinite recursion and showed that they cannot be regarded as one-dimensional lines because they pass arbitrarily close to other points along themselves as they twist and turn, so almost filling the whole of the bounded region they occupy. They are characterised as having a fractional dimension, a novel concept, quantified by Hausdorff dimension which measures the capacity of the fractal curve to fill space. Values will be between 1 and 2. One of the pioneers of fractal trees in computer graphics was Peter Oppenheimer of the New York Institute of Technology[3]. He included modelling the tree bark and leaves.

Models of a single bare tree using fractals conjecture that each section of a tree, from trunk to twig, is similar in shape to all other sections, varying only in scale and random detail. The easiest fractal tree shapes to generate are 2D silhouettes, which I now illustrate.

[3] 'Real time design and animation of fractal plants and trees.', Computer Graphics, Vol 20, 1986, p55

## Silhouettes of bare trees

I list below in Figure 10 the code in BBC BASIC which will produce the 2D symmetric bare trees in Figure 9. This is the simplest case. The code in any other language will be much the same. It consists of these statements:

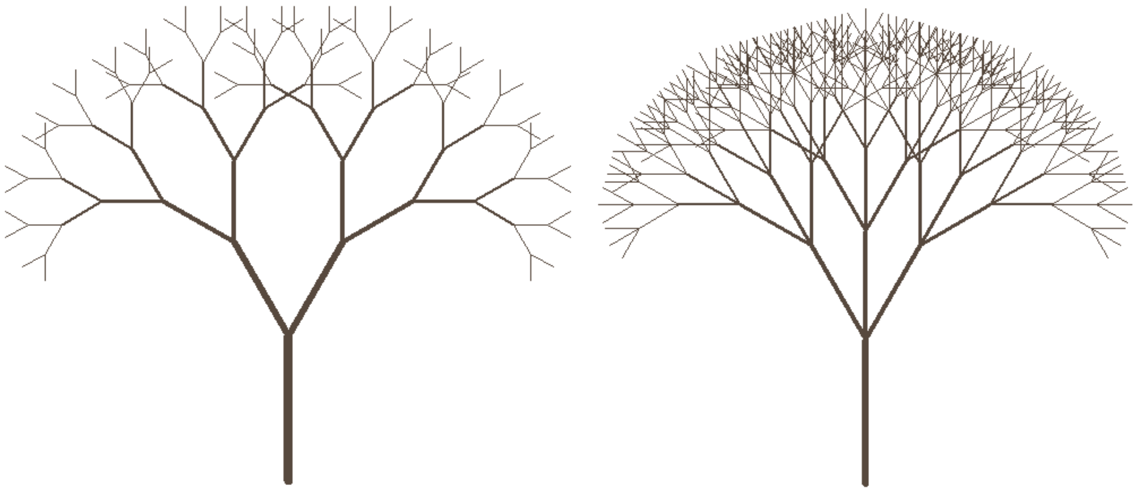
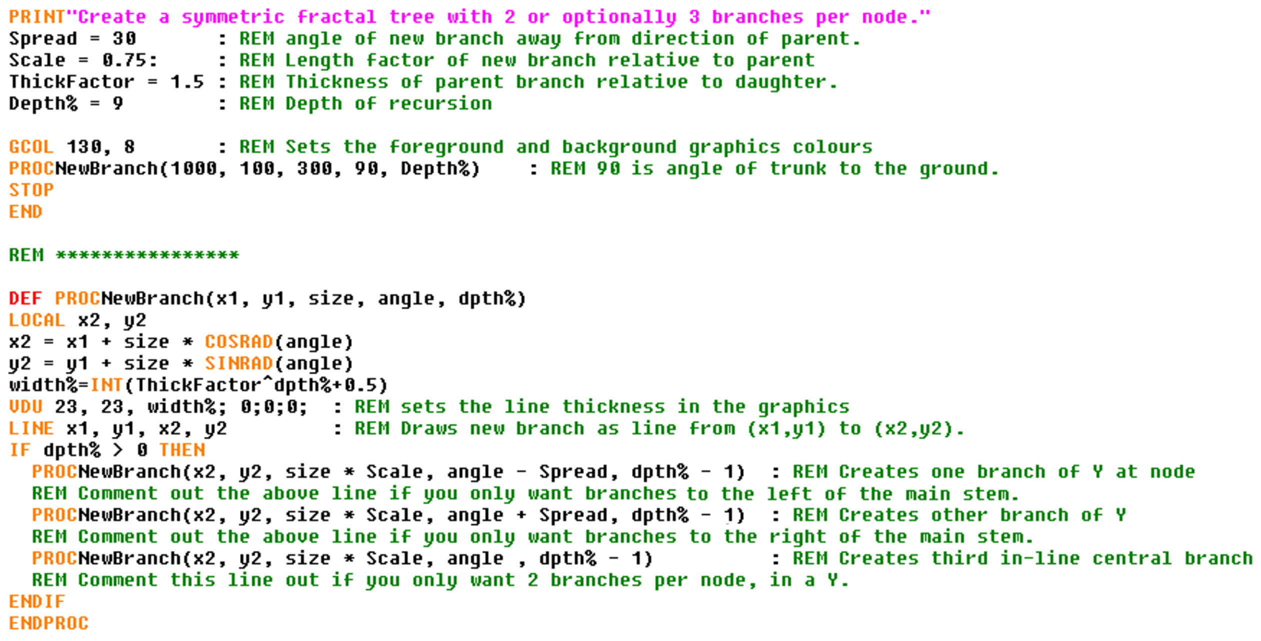


Figure 9: Highly regular tree-like fractal with two branches (left, 6 recursions) or three (right, 5 recursions) at each node. The apical dominance ratio is 0 in the left panel and 1 in the right.



```
PRINT"Create a symmetric fractal tree with 2 or optionally 3 branches per node."
Spread = 30        : REM angle of new branch away from direction of parent.
Scale = 0.75:      : REM Length factor of new branch relative to parent
ThickFactor = 1.5  : REM Thickness of parent branch relative to daughter.
Depth% = 9         : REM Depth of recursion

GCOL 130, 8        : REM Sets the foreground and background graphics colours
PROCNewBranch(1000, 100, 300, 90, Depth%)    : REM 90 is angle of trunk to the ground.
STOP
END

REM ****************

DEF PROCNewBranch(x1, y1, size, angle, dpth%)
LOCAL x2, y2
x2 = x1 + size * COSRAD(angle)
y2 = y1 + size * SINRAD(angle)
width%=INT(ThickFactor^dpth%+0.5)
VDU 23, 23, width%; 0;0;0;  : REM sets the line thickness in the graphics
LINE x1, y1, x2, y2         : REM Draws new branch as line from (x1,y1) to (x2,y2).
IF dpth% > 0 THEN
  PROCNewBranch(x2, y2, size * Scale, angle - Spread, dpth% - 1)  : REM Creates one branch of Y at node
  REM Comment out the above line if you only want branches to the left of the main stem.
  PROCNewBranch(x2, y2, size * Scale, angle + Spread, dpth% - 1)  : REM Creates other branch of Y
  REM Comment out the above line if you only want branches to the right of the main stem.
  PROCNewBranch(x2, y2, size * Scale, angle , dpth% - 1)          : REM Creates third in-line central branch
  REM Comment this line out if you only want 2 branches per node, in a Y.
ENDIF
ENDPROC
```

Figure 10: Code in BBC BASIC showing the essential recursion algorithm.

1. parameters governing the shape of each fork in the tree, and the depth of the recursion. The parameter `Scale` reduces the length of daughter branches as stated in the `REM` comments.

2. the first call to the procedure NewBranch (in other languages called a module or sub-program) which will draw the given elementary form on the graphics screen according to given input values. The first call draws the trunk, and in the example the bottom of the trunk is at position $(1000, 100)$ pixels, and its height is 300 pixels at an angle of $90°$ to the horizontal.

3. the code of NewBranch which calculates the end point of the two or three new branches and draws them from the endpoint of the trunk or previous branch. The lines have thicknesses which depend on the level of the recursion, to give the illusion of branches getting thinner higher up the tree. The angle of these in this example is $30°$. One branch is to the left, the other to the right and the third central. Optionally one of these branches can be commented out as a REM statement to give only two or even one branch per node.

4. Critically, NewBranch then calls itself Depth% times to create the smaller branches and twigs along the parent branch, reducing the scale at each call.

The recursion level dpth% is reduced at each step so that recursion terminates after a few iterations. (In BASIC % denotes an integer value.) The command LOCAL is important; it means that the listed variables are restricted to that particular call to NewBranch, and hence that a new set will be given values each time NewBranch is called.

As trees the images in Figure 9 are highly abstracted, though in art they might find a place on greetings cards, or in wallpaper or fabric designs. From such a simple, highly regular and symmetric basis more realistic tree images can be created by varying the parameters from iteration to iteration. The next stage would be to make the tree three-dimensional,

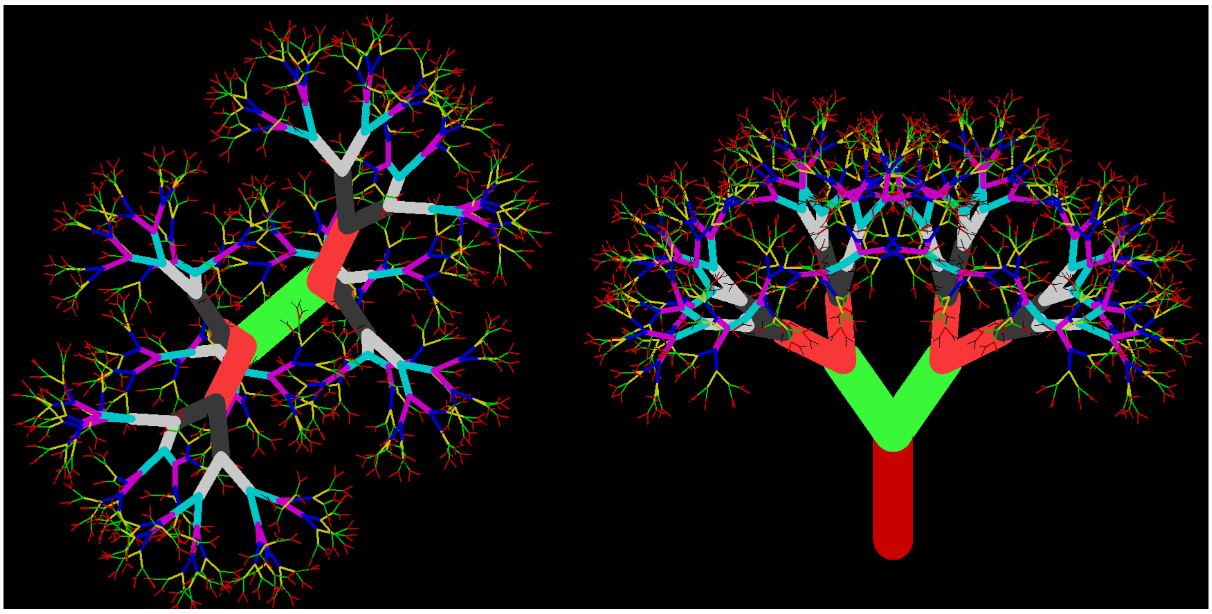


Figure 11: Regular fractal tree with two branches per node at $35°$ to its parent, with azimuth advancing $+40°$ at each iteration for 10 iterations. Views from $45°$ right (south-west) and from above. Each iteration is coloured differently.

even though it is seen only as a 2D silhouette. This can be done by introducing the angle of azimuth, $\phi$ in spherical co-ordinates, at which a new branch grows relative to its parent. The mathematics of this is more complicated so I have placed it in the Appendix. Figures 23 shows four views of the regular tree created with the code at Figure 24 in the Appendix. Essentially it takes the tree on the left of Figure 9, with two branches per node, and turns each new branch by 40° clockwise about its parent.

The next step towards realism is to introduce randomness into the generating code. I say something of this in the Appendix. The optional variations are many, but perhaps the most significant is the apical dominance ratio. Figure 26 in the Appendix looks like a young tree. Figure 12 below looks like an oak or other spreading deciduous tree without its leaves with mean ratio 1. Figure 13 is conifer-like with a higher apical dominance ratio of 2. In each case the three views are projections of the same 3D tree onto orthogonal planes, as if looking northwards, from the south-west and from the west, plus down from above. Another example is Figure 19 in Appendix 1 which emulates leaves on an old tree which has lost some of its branches. It would be an interesting challenge to reproduce these fractal algorithms in a 3D modelling package such a blender, replacing the 2D silhouette lines with 3D cylinders, and then viewing the tree, as if solid, from any position.

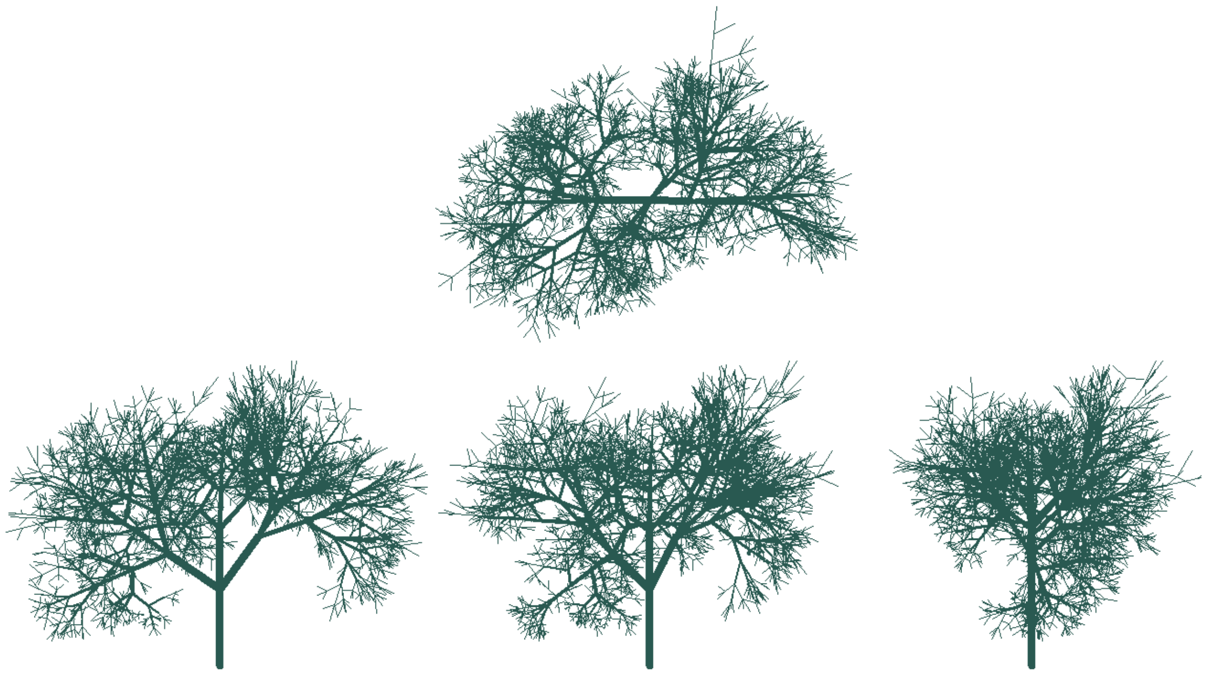


Figure 12: Three views from the ground and the view from above of a randomised fractal resembling many deciduous trees in winter.

These figures and many others like them are evidence that randomised fractal models can produce fairly realistic bare tree shapes. There is some of the biology of §2 in the algorithm through the self-similarity of branching and the choice of parameters such as apical dominance ratio, branch length and thickness, and the number of branches from each node, and the angle at which they grow from their parent. However, they are not a model of growth since they do

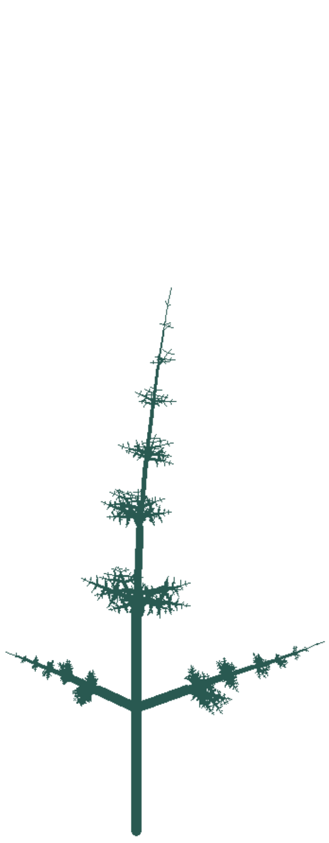


Figure 13: Four views of the same randomised fractal resembling a conifer. Polar angles are spread about 75° and the azimuthal increment is spread about 90°.

not track the growth of an individual plant from a sapling to old age. Instead these model just present one set of silhouettes at a time, with no link to the next run of the program. Moreover, they take no account of environment, which does affect growth significantly. Clearly they have none of the surface detail seen at close to. In computer art they are probably best used for generating distant trees on the skyline in low budget scenes.

## 4 Lindenmayer's L-systems

We now briefly consider a model of plant growth through time, as opposed to fractal models which generate only single snap-shots. The biologist Aristid Lindenmayer developed his model of cell growth, outlined below, in 1968 after studying filamentary algae[4]. He was describing how complex forms can develop from a few cells by repeated application of simple rules for cell division. Later he and other workers generalised and extended the idea to simulate growth of much larger, more complicated plant structures. His theory does much to explain the fractal nature of many plant forms. The comprehensive and well illustrated book 'The Algorithmic Beauty of Plants' by Prezemyslaw Prusinkiewicz and Aristid Lindenmayer, publ. Springer 2004, is available as an e-book on the internet, so here I just outline the concept.

[4] A Lindenmayer. 'Mathematical models for cellular interaction in development'. J Theoretical Biology, 18:280-315, 1968.

To illustrate the concept start with a single cell A. In any given growth cycle, whenever A appears in a sequence of cells it divides into a longer string, say A → BCA. Suppose also that B divides to AB. Over four growth cycles (iterations) the organism grows as follows:

$$\text{A} \rightarrow \text{BCA} \rightarrow \text{AB C BCA} \rightarrow \text{BCA AB C AB C BCA}$$
$$\rightarrow \text{AB C BCA BCA AB C BCA AB C AB C BCA}$$

This looks like a filamentary algae and already two larger units of the organism have emerged, 1 = BCAABC and 2 = ABCBCA, so that after the first cell division the organism consists only of unit 2, but after the third is 12, and 2112 at stage four. Here is stage four with typeface changed to emphasise the recurring structure: *ABCBCA* **BCAABC BCAABC** *ABCBCA*. By stage 5 the structure has become 1221 2112 which can be grouped in larger units as PQ. Therefore at stage 6 the structure is 2112 1221 1221 2112 = QPPQ. The self-similarity is clear. In this system there will be $3n$ cells after the $n^{th}$ division.

If the first cell to divide were B, the sequence of cells becomes rather different and a different organism is created:

$$\text{B} \rightarrow \text{AB} \rightarrow \text{BCA AB} \rightarrow \text{AB C BCA BCA AB}$$
$$\rightarrow \text{BCA AB C AB C BCA AB C BCA BCA AB}$$

After two divisions the structure in larger units is 23 where unit 3 is BCAAB, the same as unit 2 with the final C deleted. At stage 4 the organism is 1223. There are $3n - 1$ cells at division $n$. The recursive, fractal structure of organisms growing from cells A and B is apparent. In contrast cell C is sterile.

Two- and three-dimensional L-systems can be turned into graphics if one of the replacing 'words' contain the instructions to draw a line then turn through a specified angle. The process has been likened to the movement of a 'turtle' educational robot which on each instruction can advance by an integral number of steps in a specified direction. Starting from a short statement (the axiom), repeated nesting of a set of short elementary instructions (the rules) can create many attractive fractal patterns, including the space-filling curves mentioned above. Python has an implementation of turtle graphics. In this way L-systems have self-similar structure and are a crude model of growth through an indefinite number of stages in a plant's life. In graphics they have been most successful for modelling ferns, fronds and similar small plants. Many are illustrated in the book by Prusinkiewicz and Lindenmayer and on the internet together with their axioms and rules. I present four in Figure 14 which I created by modifying python code obtained from the internet. They do have a delicate grace and charm.

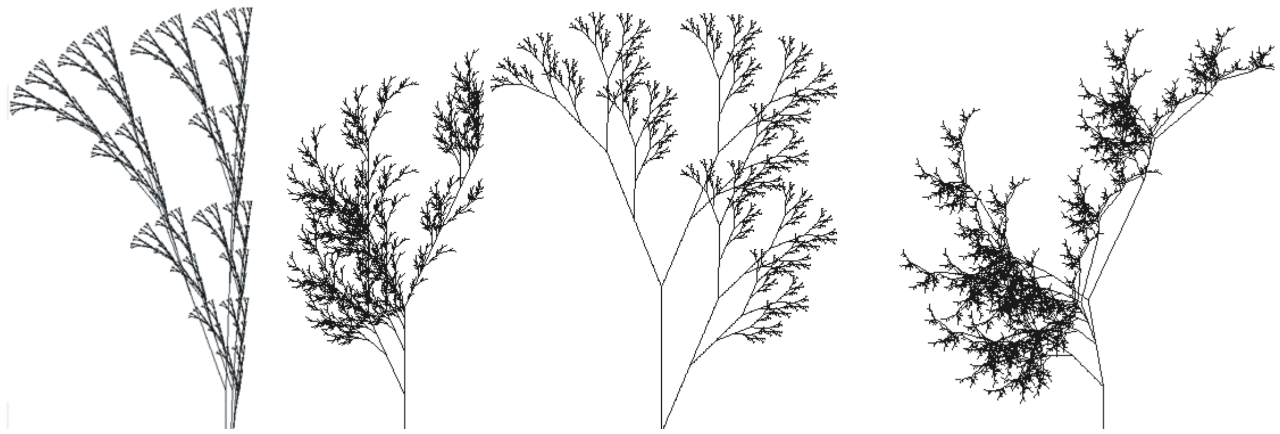


Figure 14: Four plant-like forms produced by instructing a ‘turtle’ to draw according to an L-sytem.

## 5 Physiology- and environment-driven models

Though L-systems describe a type of organic growth, we turn now to look briefly at the vast amount of research and development since about 1980 into much more realistic and sophisticated models collectively called ‘functional-structural plant models’, FSPM. Such has been the growth in this aspect of computational biology that new learned journals and international conferences have been instituted to cater for it. The developments were mainly for estate woodland management, forest planting and felling, ecology, agriculture and fire control of woodlands, not for the computer graphics industry which at the time was in its infancy.

FSPM involves regarding the tree or other plant as having about a dozen distinct biological systems which are crucial for its growth. These include sugar production in the leaves, transport of sugar, transport of water, transport of hormones, production and activation of buds, development of wood for strength. These systems take water and nutrients from the soil, oxygen and carbon dioxide from the atmosphere, starch from internal stores, and sunlight, and do so in the presence of obstacles such a buildings, and in competition with other trees nearby. The model integrates all these processes and determines the tree’s growth in size and shape in accordance with the availability of essential environmental resources, particularly sunlight. These models run in a yearly cycle so that the condition of the tree after one year will determine its capacity and trajectory for the next.

Let me attempt a simplified illustration of an idealised functional-structural plant model. Suppose that on biological grounds the tree can be divided into a small number of active components each with a structure (‘architecture’) and physiological function. For instance they might be

1. leaves to respire and produce sugars by photosynthesis,
2. roots to gather water and minerals and to store starch,
3. twigs with buds to initiate new growth, open as leaves and flowers, and release hormones governing growth,

4. a trunk, branch and twig structure to support the leaves in their search for sunlight, transport water, nutrients, sugars, starch and hormones, and give mechanical strength.

Each of these components might be regarded simply as a 'box' with input, output and control ports, by analogy with components of an electrical circuit. Rules are specified for how each box converts its inputs to outputs, perhaps in the form of a logic tree and/or a difference equation placed inside the box. An initial state is defined, representing spring awakening after winter, and the model set into action. At every increment of activity the flows amongst and within the boxes are self-regulating. Changes take place in the state of each box and in the pipelines which join them, and so the whole system in incremented. The whole system steps through hundreds of time increments over which it predicts that leaves will open, produce sugars and later die and be shed, that the tree will grow in height, mass and complexity, and that starch will be stored in the roots and branches ready for the next year. The system organises its own next steps. At each stage quantitative results are output and may be displayed as an image of the plant growing.

This illustration has similarities with network models used in systems engineering. The system can be drawn as a mathematical graph of vertices and directed edges. The boxes at the vertices are not 'black boxes' whose internal workings are unknown, but 'glass boxes'; the user understands what is going on inside and may have reliable numerical values for the various inputs, outputs and controls obtained from field measurements. The same approach can be applied to modelling many trees together in a plantation or natural forest, and to fields of crops.

As might be expected, such a self-interacting model would be ambitious in scale and complexity. Indeed, some tree and forest models have taken years to develop and refine. They explain *why* trees have the shapes and sizes they have and predict how they will change over time according to environmental conditions. In the main they have been developed by organisations with responsibility for managing trees, but some as a by-product produce convincing pictures of simulated trees, and the realism of the 3D trees they do generate is a strong measure of their correctness.

In this short article I cannot go into more detail as the number and variety of FSP models is too large, and each is too technical for ready summary. The interested reader will gain a flavour of this broad, multidisciplinary subject by referring to original technical papers of which the following are a sample. Most are available on the internet. First I mention these reviews and large projects:

- A book was published in 2008 edited by Jan Vos and others called 'Functional-structural plant modelling in crop production', published by Wageningen University and Springer. Its 22 chapters have contributions from leaders in the field at that time. Its abstract reads as follows: Functional-structural plant models (FSPMs) describe in quantitative terms the development over time of the 3D structure of plants as governed by physiological processes and affected by environmental factors. FSPMs are particularly suited to analyse problems in which the spatial structure of the plant or its canopy is an essential factor to explain, e.g. plant competition (intra-plant, inter-plant, inter-species) and the effects of plant configuration and plant manipulation (e.g. pruning and harvesting) on yield and produce quality. This book describes the philosophy of functional-structural plant modelling and several tools for making FSPMs. It outlines methods for measuring essential parameters including those pertaining to plant structure.

- A wide review of plant modelling, covering L-systems, FSPM, and CGI models, has been given by Fumio Okura of the University of Osake computer vision department in 2022, published as '3D modelling and reconstruction of plants and trees: A cross-cutting review across computer graphics, vision, and plant phenotyping', Breeding Science Vol 72: p31-47, 2022, published Japanese Society of Breeding. He categorises modelling into i) creating non-existent plants, and ii) reconstructing real plants from images and measurements in nature. This fairly recent review is valuable for showing how the many models stand in relation to one another, and for its extensive references. The review deals mostly with models of plant shape and structure.
- GreenLab is a FSPM which has been in development for 20 years. It has been reviewed by Wang, de Reffye and others in Plant Phenomics, Vol 6, 2024. doi:10.34133/ plant-phenomics.011. The review describes these developments and appends a lengthy list of references. The developers say that GreenLab is a mathematical dynamic model aiming to model and simulate plant structure establishment and production. It differs from computational models by the fact that both development and functional processes are described by equations. The model therefore quantifies structure (the number of organs, etc.) without requiring exhaustive structural implementation. It also differs from previous FSPMs in that organ production is quantified by compartments competing for a common biomass pool. In a growth cycle the model sequences organogenesis, biomass production and its partitioning in a dynamic loop.
- OpenAlea has been an open source project for the plant research community drawing on collaborative effort to develop python libraries, running in Linux, related to plant modelling. The project has a site on GitHub which includes modules to analyse, visualize and model the functioning and growth of plant architecture. The project appears still to be active. It was announced back in 2008 in 'OpenAlea: a visual programming and component-based software platform for plant modelling' by Pradal, Godin and others from France, published in Functional Plant Biology, 2008, 35, 751-760. The project includes a graphics tool kit called PlantGL developed in France; see Christophe Pradal *et al.* 'PlantGL: A Python-based geometric library for 3D plant modelling at different scales' in Graphical Models, Vol 71, 2009, p 1-21. Pradal is still active in plant biophysics and modelling.

A sample of more specific papers in date order is:

1. Of the early studies regularly cited in the literature are joint papers by Jack Fisher from Florida and Hisao Honda from Japan. From the early 1970s they studied branching and leaf area of semi-tropical trees. A paper from 1982 reporting a study of branching patterns is representative: 'Two geometrical models of branching of botanical trees', Annals of Botany, Vol 49, 1982, p1-11.

2. Two organisations which have been active in tree modelling for arboricultural purposes for many years are the University of Helsinki and CIRAC, the French agricultural research and cooperation organization working for the sustainable development of tropical and Mediterranean regions. Philippe De Reffye is the research director of CIRAD. Back in 1992 he co-authored with Marc Jaeger a paper on 'Basic concepts of computer simulation of plant growth', J. Biosci., Vol. 17, 1992, p 275-291. It assesses the botanical basis of plant architecture. Emphasis is given to the buds as sites of new growth. Their activity is simulated by a Monte-Carlo process in which random numbers are combined with stochastic laws of new growth, branching and self-pruning. They apply their model to several semi-tropical plants using field measurements for accurate qualitative data on the architecture/geometry of each species and its growth pattern. Figure 15 is taken from this 1992 paper and shows four stages in the growth of a cherry tree.

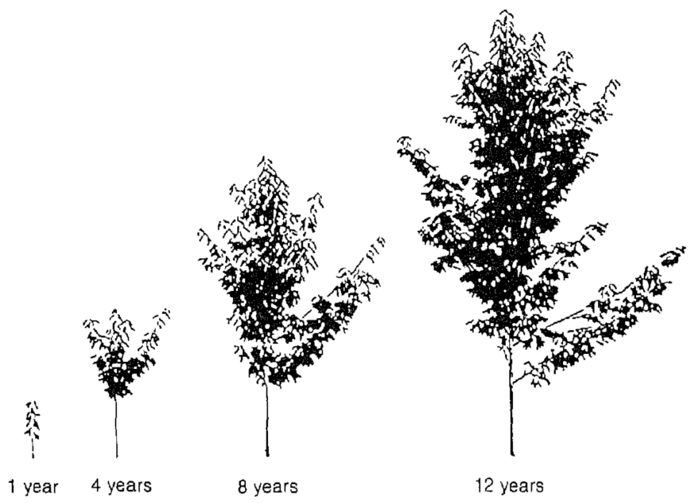


Figure 15: Growth of a wild cherry tree as modelled by Jaeger and de Reffye, 1992.

3. Several authors have focused on the cardinal importance of available sunlight for leaf growth. In 1994 Takenaka of the Japanese Institute for Environmental Studies published 'A simulation model of tree architecture development based on growth response to local light environment' in J. Plant Research, Vol 107, 1994, p321-330. He built on Honda's model of branching from the 1970s by incorporating the amount of light falling on the branches. The unit of tree architecture is a stem he called the 'branch unit' which has leaves at the terminal node. He calculated the local light level taking into account mutual shading among leaves. As the tree grows, the number of leaves increases and mutual shading becomes intense, so that fewer new branch units are produced and others are shed to restrain overcrowding. He simulated changes in the shape of the tree canopy over the years and found that the predicted shape of tree crowns was similar to that in real forests.

4. L-systems have retained their appeal to several workers, with attempts being made to combine them with biological processes. One example of this, applied to a birch tree, was published in 2005 by Michael Renton and colleagues in a joint Australian-Finnish collaboration as 'Functional-structural plant modelling using a combination of architectural

analysis, L-systems and a canonical model of function', Ecological Modelling, Vol 184, 2005, p 277-298. Though they use mathematical functions to describe the physiological dynamics, these are empirically derived from observations of actual trees. The authors see this as a benefit in that it is relatively simple compared to process-based FSPMs and does not require a detailed understanding of underlying physiological, yet is able to capture important aspects of plant function and adaptability. One conclusion from their results is that using light as the sole factor to determine the structural location of new growth gives satisfactory results. I think that today such a phenomenological way of treating the physiology would no longer be used; the paper represents the technical position of its time.

5. LIGNUM was a FS tree model developed by Jan Perttunen while a graduate student at Helsinki University in 2008. It incorporates many physiological process including sunlight distribution, respiration, allocation of sugars and starches, fluid transport through the tree, and patterns of bud and branch growth in length and thickness, expressing dependencies as mathematical formulae. Perttunen makes limited use of L-systems. He applies his model to both conifers and deciduous trees. The model was reported by Perttunen and colleagues as 'Functional structural plant models - case LIGNUM', doi:10.1109/PMA.2009.64. Figure 16 is taken from their paper.

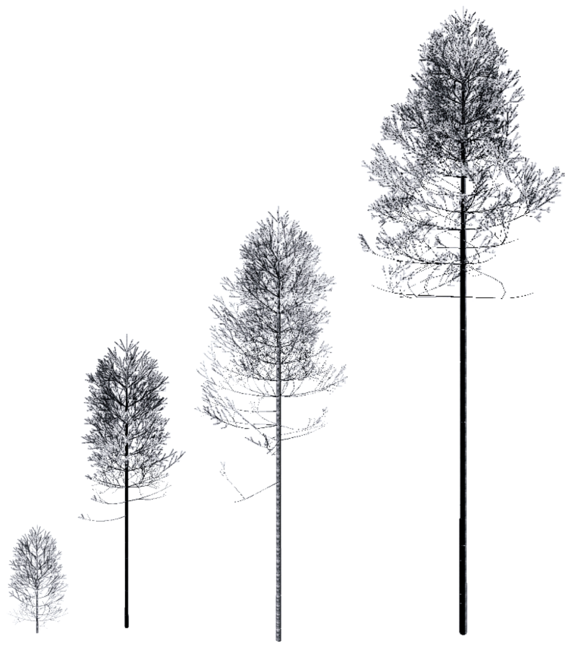


Figure 16: Images from Lignum of a simulated Scots pine tree at ages of 10, 20, 30 and 40 years, growing to 12m high.

6. Much experimental data on trees, copses and forests is from laser scanning, either from the ground or the air. One approach is to scan the tree in winter when only the branches and twigs are present, and infer the leaves' contribution to total biomass. Akerblom and others in a multi-national team have developed an algorithm to insert leaves into such scans of bare trees to obtain plausible leaf area density as it is distributed over the crown

of single trees and the canopy of a copse: 'Non-intersecting leaf insertion algorithm for tree structure models', Interface Focus, Vol 8, Royal Soc. 2018. This paper therefore focuses on clusters of tree leaves. The authors outline a MatLab implementation available at *github.com/InverseTampere/qsm-fanni-matlab.*

7. Yi, Li, Zhang and colleagues at Xi'an University, China, working with European researchers, have developed a tree growth model directed towards CGI but with a strong grounding in plant biology. 'Tree Growth Modelling Constrained by Growth Equations' published in Computer Graphics Forum, Vol 37, 2018, p239. doi/10.1111/cgf.13263. Figure 17 reproduces their figure 11, showing the model's success in growing the tree to avoid a wall. They call their approach 'integrated growth modelling'; morphology is integrated into growth equations with selectable parameters setting sensitivity to light, allocation and usage of input resources, and environmental. The growth equations ensure that the simulation numerically matches the natural growth of real trees, including in the presence of obstacles and other competing trees. They list seven stages in their model:

   (a) Generation of the initial trunk
   (b) Calculation of growth according to the light falling on the leaves and allocation of water and nutrients,
   (c) Refinement of growth by the logistic growth equation as cited in B. Zeide's 'Analysis of Growth Equations,' Forest Science, Vol. 39, 1993, pp 594-616. The authors quote the formulae they use.
   (d) Append new shoots at every sprouting node in random order through a space colonization algorithm which takes account of the light falling there,
   (e) Calculation of environmental impact of new shoots – light level and physical impact with adjacent structures,
   (f) Calculation of branch radii and the shedding of branches,
   (g) Generation of structure data for finishing this current annual growth phase.

   When a clump of tree are simulated, the trees show individually different shapes according to their position in the clump. The main missing ingredient is the effect of gravity on

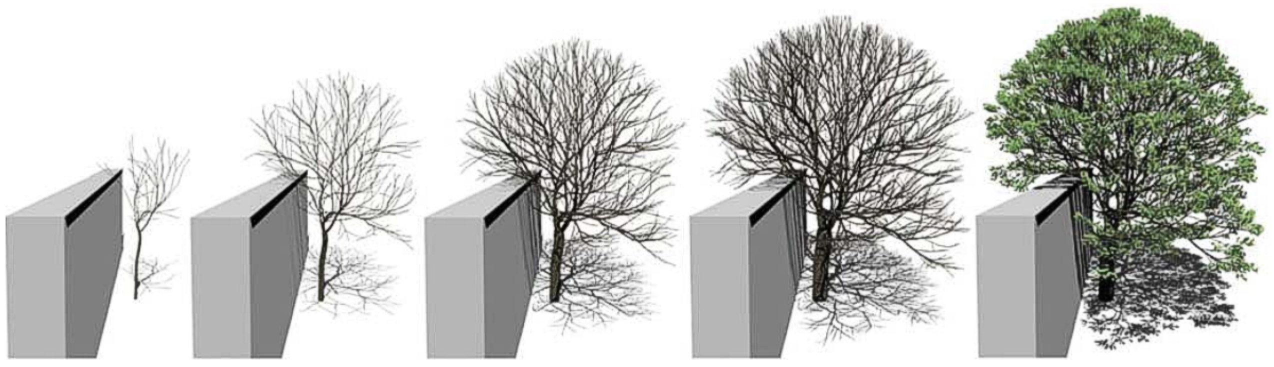


Figure 17: Figure from Yi *et al* modelling tree growth next to a wall.

branch strength, bending and self-pruning. There are strong similarities with two CGI models, The Grove and Natsura, described in the next section.

8. As an example of the FSPMs still being developed I cite the joint work from Hungary and Slovakia of Tóth and Szénási reported in 'Tree growth simulation based on ray-traced lights modelling', Acta Polytechnica Hungarica, Vol 17, 2020. They describe a fairly typical functional-structural model to which their main contribution has been using ray tracing from the sky and through the leaves to determine the light strength on leaves across the whole tree. As ray tracing to each leaf would be too computationally intensive, they have simplified the calculation by a Monte Carlo algorithm, though they do not describe it in detail. They take the direction of growth to be the direction from which comes the strongest light over the growing cycle, though they do not mention the sun moving across the sky each day. The model runs over several annual growth cycles, and they comment on the increasing computing power required year on year as the number of leaves grows rapidly. They therefore advocate the parallel processing strengths of a modern graphics card (GPU). I do not consider their work an advance on what others had done previously, but cite it to illustrate the range of approaches and the detailed aspects various workers have addressed.

9. Hans Pretzsh, 2021, 'Tree growth as affected by stem and crown structure'. This is a study related to 'metabolic scaling theory' which relates metabolism and growth to mass. Established semi-empirical rules such as Kleiber's law (for plants also called the West-Brown-Enquist law) state that the rate of biomass production varies as the $\frac{3}{4}$ power of a tree's mass. Pretzsh's long term study analysed the effect of structural and morphological tree characteristics, such as crown volume and previous growth rate, on the growth of 1600 common European trees – oak, beech, spruce and Scots pine – aiming to improve on Kleiber's law for stands of mixed tree species as used in reforestation and sustainable forest management.

10. A Japanese paper from 2022 by Tsugawa and others has examined the mechanical strength and flexibility of trees using finite element analysis: 'Exploring the mechanical and morphological rationality of tree branch structure based on 3D point cloud analysis and the finite element method', published on-line in 2022 by Nature Scientific Reports. They applied their analysis to actual larch trees previously measured by laser scanning. They conclude that the maximum stress depends on the branch inclination angle to its parent, and that branches change their growth direction, and therefore become curved, to limit the maximum bending stress. These results are hardly surprising, but this study has quantified the subject and suggested that different tree species have different mechanical strategies which determine their shapes.

11. Plants grow with the process of phototropism, which is their propensity to extend and curve towards the source of light to maximise the capture of sunlight by the leaves for photosynthesis. This aspect of growth was reviewed in 2024 by Tristan Nauber and Patrick Mäder of Germany: 'Light distribution models for tree growth simulation', Computer Graphics Forum, Vol 0, e15268, pub. Eurographics. This is part of international research and development into functional-structural plant modelling.

## 6 Some commercial software for modelling trees

CGI trees are used in computer games and in cinematography as components of a virtual world. The sheer complexity of trees, especially when seen close to, poses a major challenge to computer representation. Even today making 3D trees and plants is still difficult, partly because of from their form and partly from the need to render then to screen quickly. In most software 3D forms are represented by a mesh of vertices, with the faces connecting them given the appropriate colour, projected image and/or texture. One early technical paper which describes in some detail the considerations behind the creation and rendering of realistic trees is by Jason Weber and Joseph Penn of the University of Florida[5]. When viewed close to, 3D tree models require a mesh of many thousands of polygons to conform to their gnarled surfaces and branching form. Normal and displacement maps can provide the rendered image with the illusion of small height variations in the surface, so can produce convincing tree bark and moss without increasing the polygon count. Depicting leaves[6] is another challenge. It has been addressed in several applications by painting each leaf onto a rectangular surface, and making the outer part of the rectangular, outside the leaf margin, transparent using the alpha channel of the colour. The industry calls this 'transmapping'. At rendering these parts behave as if invisible so we see only the leaf and its correctly cast shadow. In addition the leaf material itself can be given a partial transparency similar to real leaves.

Trees pose a further challenge in computer games, where the action is in real time and scene re-rendering is at a frame rate of 30 per second or faster. This is demanding of the graphics hardware so a balance must be struck between keeping the number of polygons to be rendered small whilst maintaining an acceptable level of realistic detail. Controlling the level of detail, 'LOD', is a major algorithm in games engines. The concept is to reserve rendering a finely divided, high density mesh for objects close to the camera, and to reduce the number of rendered polygons by coarsening the mesh for objects at increasingly great distances. The algorithm may reduce the LOD through three or four discrete levels, or do so in a continuously as range increases. The lowest stage for distant trees may be the plainly coloured convex hull of the tree, or even just a 2D image called a 'bill-board', placed normal to the line of sight. A subsidiary challenge is to degrade the resolution subtly to avoid trees suddenly and noticeably switching appearance as the camera zooms out – so called 'popping'.

A related challenge for CGI is the requirement to render a large number of trees and similar plant objects in the scene. This is addressed by 'instancing' in which multiple copies of a tree are rendered simultaneously with more or less the same material. Each copy is called an 'instance'. To add variation and limit obvious repetition each instance can be given a different colour and/or scale. The graphics processor treats the call to draw the instances as one batch process, so restricting the strain on its computing power. It is important that the trees in a scene can be rendered to an acceptable level of detail in a time compatible with the movement of the camera and swaying of the tree itself in the (virtual) wind.

[5] Proceedings of the 29th International Conference on Computer Graphics and Interactive Techniques. On-line at https://dl.acm.org/doi/10.1145/218380.218427

[6] An analysis of light interaction with leaves in rendering was given by Wang and others in 2005 : 'Real-time rendering of plant leaves'. https://graphics.cs.yale.edu/sites/default/files/leaf2005.pdf

This section is a non-exhaustive catalogue of computer packages which provide trees and foliage for inclusion in a scene. Many run on a home computer under Windows, Mac or Linux. Some are stand-alone programs, some add-ons to modelling platforms such as Blender and Daz Studio, and some are closely linked or incorporated into comprehensive landscape and environment software platforms such as Houdini and Unreal Engine. A distinction can be drawn between applications which are tool kits for an artist to construct a geometry resembling a tree, and ones which attempt to follow nature and grow the tree in simulation of biology, though there is a spectrum between these extremes. I have not been able to discern the extent to which the various tree generating programs are founded on plant physiology; most appear to depend on visual appearance. Some of the more successful modelling programs seem to involve a mixture of photogrammetry of actual trees to create a mesh with overlaid texture, then modified for the particular platform to optimize easy of use and speed of rendering.

Most models are written in C++, JavaScript or Python and are said to be ‘procedural’, meaning that they have in-built algorithms with user-adjustable parameters, each of which will generate a geometrical or textural element of the tree. These algorithms govern critical aspects according to in-built rules, and the user customises the tree subject to these rules. In this way the tree and its components fit and function together automatically, something that would be most difficult to achieve if every detail defining the tree had be be crafted by hand.

At the time of writing two of the most sophisticated software engines for creating realistic environments and special effects are Houdini by SideFX Labs and Unreal Engine 5 by Epic Games, known in the industry as UE5. Houdini has probably been used more for motion picture landscapes and moving special effects such as fluid flow, fires, wind motion, whilst UE5 is more for development of computer games in which heroes and villains roam virtual words in search of treasure, fame or vengeance.

In Houdini the user interface (UI), from its first version, has been designed to make its procedural ethos clear to the user. Each procedural operation will run an algorithm – for example, to create a new object, scale it, add a new branch of a tree to its parent, add a roughness texture or change its colour. In the user interface this operation is represented simply by an icon called a ‘node’. For each such node its user-adjustable parameters are presented as sliders or typeable numeric values. The flow of operations which creates a tree or indeed a whole scene is depicted in the UI by pipelines which connect nodes together in the required order, rather like plugging wires between components of an electrical circuit. Such a simply presented user interface makes using the software *relatively* easy because it separates the underlying algorithms (hidden behind the nodes), the adjustable parameters, and the flow of operations. Similar node-presented user interfaces have been adopted in Blender’s shader and geometry nodes, and in several other programs. Though Houdini has a free ‘Apprentice’ version, it is fairly expensive for the amateur to license a version which will export created environments.

UE5 is a powerful and comprehensive application, free to download and run. It has the tools to create virtual worlds, including landscapes with trees and plants, skies with lighting effect, buildings and artefacts of cinematic quality. Epic Games also own the vast Quixel Megascans photogrammetry library of textured 3D meshes constructed from scanned images of actual trees, rocks, buildings and a myriad other features and objects. The Quixel library is currently free and linked to UE5, so landscapes with trees, vegetation, cliffs and buildings can be created fairly quickly by composing complete textured meshes imported from Quixel. I cannot be sure how their trees have been modelled, but Epic's short promotional videos imply they use a combination of meshes and textures gained by photogrammetry of real trees married with a functional-structural model of how the tree grows and functions over its annual cycle and over its years of growth. Epic Games also have their own photogrammetry software called RealityCapture 1.5 linked to UE5, and it too is free to use by students and small businesses.

Notable features of trees in UE5 are:

- several versions of each species, including spreading ones for open fields and tall upright ones for placing together in a wood.
- multiple instances of trees can be 'painted' into the landscape with a user-held 'brush', allowing rapid population of the scene,
- leaf transparency, by which sunlight can filter through to leaves and branches below,
- in-built colour and leaf density variations with the seasons, including bare trees in winter,
- movement of the branches and leaves in the wind, the wind strength and direction being adjustable parameters,
- an innovative fast algorithm called Nanite for rendering a scene with the GPU in which the level of detail of each object in the landscape is automatically adjusted depending on its distance from the camera. Very large detailed meshes from photogrammetry of trees and plants at close range can be used since Nanite so optimises the LOD computation that it can recalculate a panned and rotating view at high frame rates. With Nanite wind-induced motion of trees can be viewed in real time.
- an illumination system called Lumen which lights the scene with reflections and shadows in real time.

There are similar libraries of scanned trees and vegetation tailored to other CGI platforms. For example Blender is served by many optimised images, meshes and add-ons for sale on Bendermarket.com, including the trees, shrubs, forests, grass, meadows, and garden hedges sold by Bproduction at *bproduction-3d.com.* Their 'Vegetation 5' add-on offers seasonal variations from bare tree to full leaf then leaf fall similar to Quixel's. Similarly several artists have crafted realistic looking trees for Daz Studio. Amongst these are the Ultra- series – UltraTrees, UltraScenery 1 and 2 – and several others. These appear to be fixed models for importing into a scene, not customisable procedural applications. Realism can be enhanced using Daz's dForce physics capability so that leaves fall and the tree sways in the wind. UltraScenery has a large enough range of trees, plants and natural objects to create a realistic scene, and in this way it emulates UE5.

Let us now leave the latest from Quixel and other modern applications and go back to a few of the earliest tree modelling programs to see how things developed. Arbaro (2012) is an implementation in Java of the tree construction by Weber and Penn noted above. It is open source and free to use from SourceForge. Though not a model incorporating dynamic physiological functions, its parameters derive from real tree morphology. Figure 18 shows three Lombardy poplars and one spreading deciduous tree which I generated using Arbaro as an `.obj` file and imported into Blender for colour to be added to the trunk, branches and leaves (the separate component groups), then lit and rendered. Arbaro divides the tree into four level: trunk, large branches, small branches and leaves. There are several adjustable parameters for each including trunk radius, curvature and taper, angle of the branches from their parent, and the number and shape of the leaves. The bare poplar trees in Figure 18 were made simply by setting the number of leaves to zero. There are about a dozen templates for common trees, bushes and grasses and each includes random components such that by choosing a different seed for the random generator a statistically different tree is generated. At the time the program was written, its graphic output was a significant advance on hand-crafted trees and tree-like fractals.

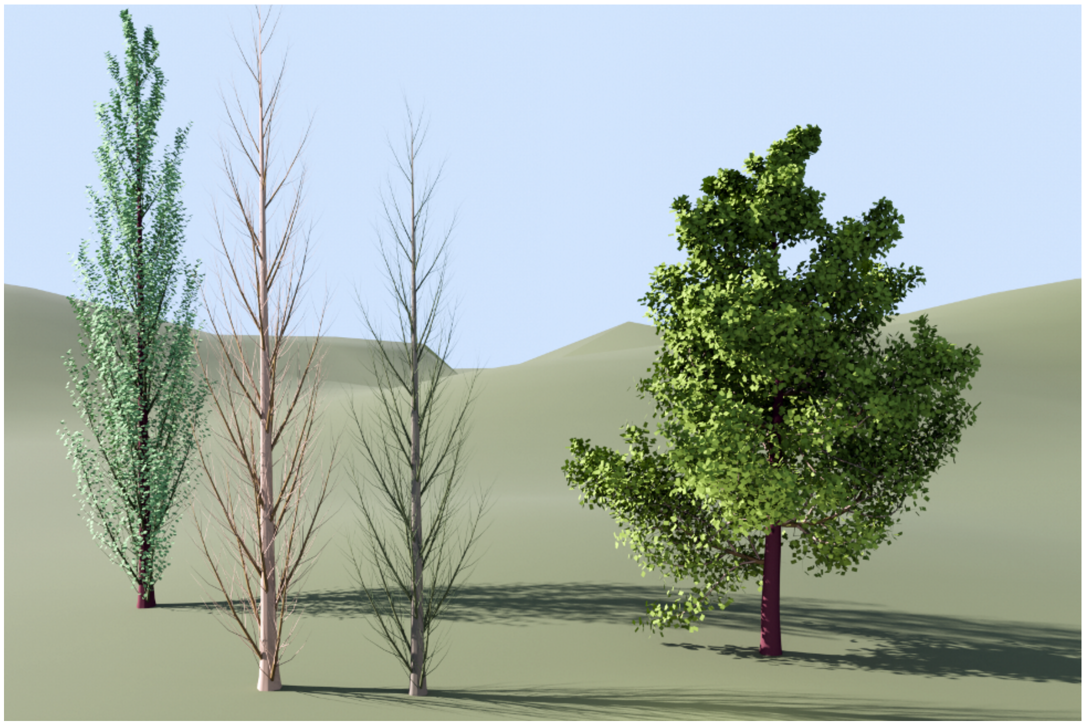


Figure 18: Trees generated as `.obj` files in Arbaro and rendered in Blender.

Another older program is ngPlant (2016), also realised in Java as WebPlant. Though this is a free, open source plant creation tool, and though some artists have made plausible plants with it, it is no longer supported and has been superseded. It has little biological basis. It is tool providing the user with the ability to build a branching structure in three or more levels – trunk, large branches, small branches and optionally leaves – with some randomness determined by a seed value. Using sliders the user steers the many parameters which determine the size, shape, orientation and number density of each level so that a wide

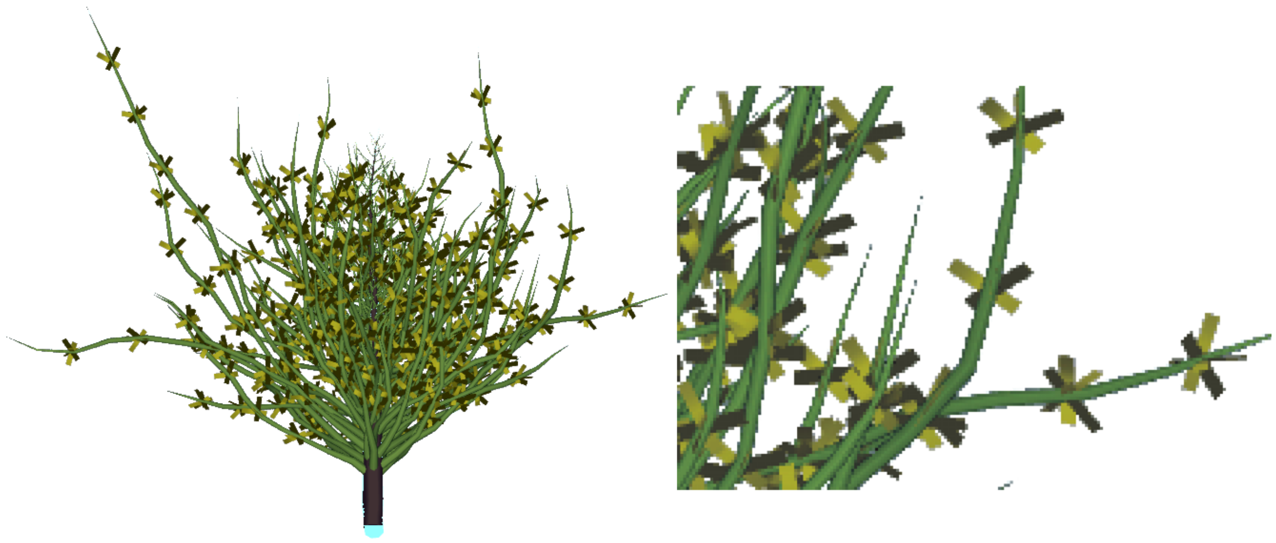


Figure 19: A scrubby bush created in ngPlant/WebPlant. Left: the bush. Right: detail showing the crude representation of twigs and foliage.

variety of structures can be readily generated. However, the branches are essential just cones stuck on the sides of their parent so the result is not particularly convincing, especially at close range. The model is 3D and can be output, like Arbaro, as an `.obj` file for import into Blender, say. I created the anonymous bush in Figure 19 with WebPlant.

'Tree It' by Evolved Software is similar to ngPlant/WebPlant in being free and open source, and having a similar user-interface. Its concept is also much the same since it too is essentially a construction kit of trunk, branches and leaves which the user customises to generate the tree-like object of their imagination. It is not built on biological models beyond the branch and leaf elements looking somewhat like real branches and leaves. Unlike ngPlant, however, Tree It has been maintained up to the time of writing, though I could not get it to run on my PC under Windows 10. It is served by some detailed tutorials on YouTube which show it behaving like an extended and more sophisticated version of ngPlant. It includes textures and normal maps for bark and leaves. It uses alpha-channel transparency with the leaves to permit a realistic leaf shape to appear on a rectangular polygon, as outlined above. The leaves can be folded along their central spine and curled which significantly increases their realism. The palm trees generated with it seem quite convincing even at fairly close viewing range. The program has proved popular with creators of computer games for which highly realistic trees are not essential. There is a library of trees made with Tree It which can be customised and randomly varied. Trees with their textures can be imported into Unreal Engine as `.fbx` files where, by giving many instances the same textures, the computing power required for rendering is contained.

Yet more sophisticated than Tree It is SpeedTree 10 by Interactive Data Visualization, Inc. The company claims SpeedTree to be the industry standard vegetation tool kit, used for both games and film. The program is indeed modern, powerful, sophisticated and highly versatile, but it is still a construction kit for the skilled graphic artist, not a biological model of tree growth and morphology. In essence the concept is similar to both ngPlant and Tree It: create a trunk, add branches at two, three or four levels, add polygons for the leaves on which are drawn the leaf shapes and textures surrounded by a transparent edge. The user interface is modern, similar to Houdini, Blender and UE5, with the tree's components being represented by procedural nodes in a network linked together by flow lines, each node with its parameters adjustable by sliders. The tree branches in SpeedTree can be curved and and twisted to make then realistic. The overall shape of the tree can be defined by an invisible enveloping shell, such that the branches and leaves come up to the envelope but do not penetrate it. By this device topiary hedges can be created. A novel innovation is to conform the plant to a target object such as an old tree trunk or a rock. This is done by making the target 'attract' the plant stems so they cling to it like ivy. The leaves in SpeedTree seem to be imported from libraries such as Megascans. A facility added in SpeedTree 9 is an interface between a) meshes of the trunk or other part created by photogrammetry and b) the SpeedTree model. The imported mesh is optimised by SpeedTree for efficient rendering. With each version this product seems to add another layer of realism, including the physics of branches drooping under gravity and the pruning of trees. In skilled hands the results are photo-realistic, though the learning curve is steep. Perhaps the high detail it gives is more appropriate to cinema and still photography than to games environments.

Houdini has a tree-building capability similar to but more modern than Tree It and SpeedTree. Trees are made fairly quickly using Houdini's node-based procedural work-flow. Trees and other objects generated in Houdini can be imported smoothly into HE5.

All the tree-generators above, apart possibly from Arbaro, are essentially based on visual appearance, not a biological model. Perhaps as a reaction to an over-dependence on the artist's input, Wybren van Keulen of the Netherlands in about 2014 set about writing software for tree growth based closely on biology. In this way he was looking back to the approaches of the 1980s and 90s when functional-structural tree models were first being written. He has probably read most of the articles in §5 and others. van Keulen called his software 'The Grove' and his company F12, and structured his program as a paid add-on for Blender and Houdini. Development has been maintained, with the version at the time of writing being 2.1. van Keulen emphasises that it is a tree growth tool, not just another way of modelling tree morphology. Each tree starts as a single shoot with buds and, with the click of the mouse, advances by one year over which it grows to a sapling. As the years are dialled up, the buds sprout new branches and leaves, many of the new branches fail to develop and fall off – a natural pruning to optimise the whole plant – and wood is added to the trunk and branches to increase their strength. Branches will droop under their weight, and are treated mathematically as beams so the deflection is physically correct. The thickness of major branches and trunk are calculated from twigs downwards, so that the lower branches can support the weight and wind load above. The software has in-built algorithms for the flow of water, sugars and hormones. It knows apical dominance, geotropism and phototropism so that a growing tree will avoid other trees and buildings nearby. The latest versions also model the root system.

van Keulen makes much of the annual natural self-pruning of the tree to prevent itself becoming choked with twigs and and leaves beneath its crown. As the branches bend under their own weight, the proximity of twigs changes and this in turn determines which twigs are shed. He has also programmed the facility to prune the tree artificially, simply by drawing a line across a branch on screen. Thereafter the tree grows as it would, with new shoots appearing around the amputated branch. Less natural is the feature to draw in new branches with a pen on the screen. I feel this is bowing to pressure from graphic artists who still wish to model tree morphology hands-on. However such artist intervention may be needed to obtain a tree which fits a particular specification – size, shape, etc. – for a game or cinema scene. With a biological simulation the growth process follows nature with a will of its own and the outcome is not wholly under the artists' control.

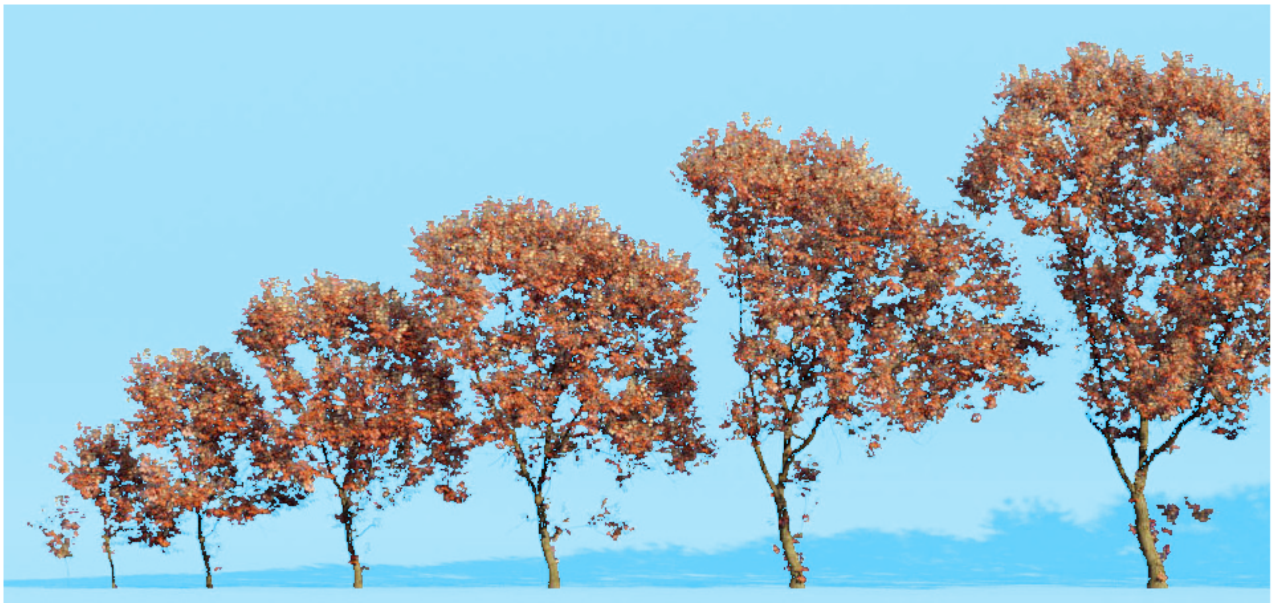


Figure 20: The Grove promotional picture.

The Grove is sold as an add-on to Blender as a core plus several optional supplementary modules each of which provides realist leaf-covered twigs characteristic of one species. In use these twigs are appended to the terminal branches. Overall The Grove 2.1 is arguably currently the most realistic software for creating 3D tree. Figure 20 is a promotional picture from The Grove's web site showing the stages of growth on one tree.

Competition is strong amongst developers. George Hulm for SideFX has recently been developing a new special tool for trees and foliage called Natsura (they also call it Natura) based on the concept of taking the best aspects of a) procedural artist's by-hand geometry creation, b) biological simulation and c) fractals and L-systems. Natsura is aimed to be an add-on for Houdini. Figure 21 is taken from SideFX's promotional video and shows how their product, though still in development, constrains a tree to grow in a constricted space. Arbaro, The Grove and Natsura could be said to embrace the principle 'Don't mimic the desired outcome, rather mimic the underlying driving phenomena'.

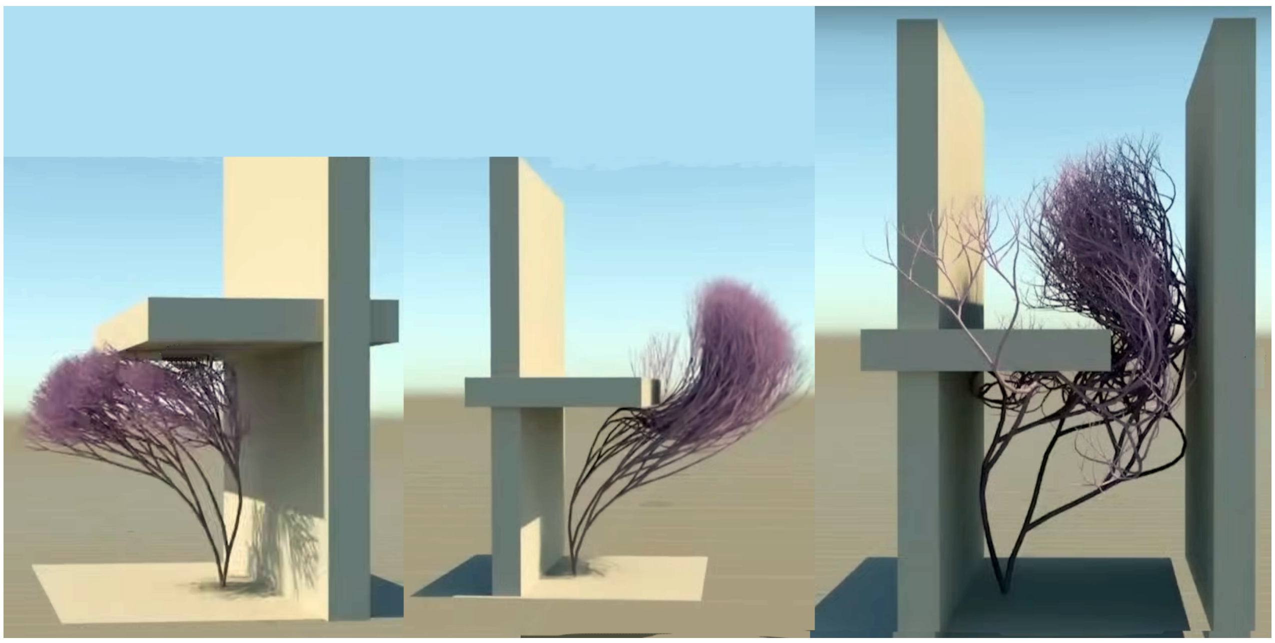


Figure 21: Three images of a tree growing between buildings as generated by SideFX's Natsura.

## Appendix: A 3D randomised fractal tree

This appendix elaborates the 2D fractal model of a tree-like structure in §3, extending it to three dimensions and giving examples of trees in silhouette.

The extension from 2D to 3D involves some vector mathematics plus the involvement of two nodes for each new branch, not just the one. A new branch can grow at any azimuth – that is, the angle around the parent branch – so we need to use a spherical co-ordinate system oriented along the parent branch in order to define the position and direction of the new branch. The geometry is drawn in Figure 22. It shows a global Cartesian co-ordinate frame $xyz$ with origin at O, the root of the tree trunk. We can take the $x$-axis as pointing east and the $y$ pointing north. At least two branches PQ and QR have already been constructed and we consider the growth of a new daughter branch RS from node R. The reason two branches must be considered is that the angular position of RS around branch QR – its azimuthal angle $\phi$ – is measured from the plane defined by the two branches PQ and QR. Of course, if PQ and QR are in the same direction, one the continuation of the other, they do not define a plane and azimuth has no meaning.

Assuming that QR is not a continuation of PR, we erect a local spherical coordinate system $(r, \theta, \phi)$ with origin at Q. The polar axis is vector $\mathbf{r}$ along the present branch. Directions around this branch are specified by $\phi$. The reference direction of $\phi = 0$ is the vector $\mathbf{q}$ which is normal to $\mathbf{r}$ and lies in the plane PQR, as indicated by the pale blue area representing that plane. A third and mutually orthogonal vector is $\mathbf{r} \times \mathbf{q}$, parallel to $\mathbf{p} \times \mathbf{q}$.

At the start of the stage of iteration to create branch RS we know the co-ordinates of

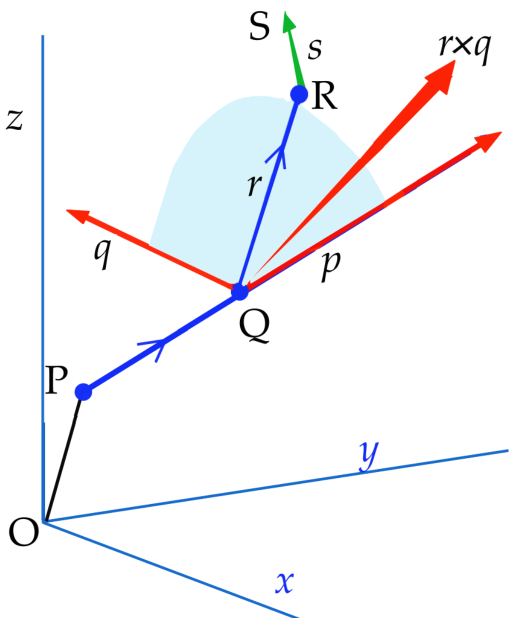


Figure 22: Vectors stating the position of nodes along a sequence of branches, and the orientation of previous branch PQ, current parent branch QR and new branch RS.

points P, Q and R. Introduce unit vectors where $\mathbf{e_p}$, $\mathbf{e_r}$ are along their respective directions. $\theta$ is the angle at which branch QR deviates from PQ and is given by $\cos\theta = \mathbf{e_p}\cdot\mathbf{e_r}$. At this stage some numerical values might be helpful. Take the example is which

$$\mathrm{P} = (1,\ 0,\ 2),\quad \mathrm{Q} = (1\cdot5,\ 0,\ 2\cdot4),\quad \mathrm{R} = (1\cdot7,\ 0\cdot3,\ 2\cdot6)$$

$$\text{so } \mathbf{p} = (0\cdot5,\ 0,\ 0\cdot4),\qquad \mathbf{r} = (0\cdot2,\ 0\cdot3,\ 0\cdot2),\qquad |\mathbf{p}| = 0\cdot64,\quad |\mathbf{r}| = 0\cdot41.$$

and branch QR is 64% the length of PQ. The unit vectors along PQ, QR are

$$\mathbf{e_p} = (0\cdot781,\ 0,\ 0\cdot625),\qquad \mathbf{e_r} = (0\cdot485,\ 0\cdot728,\ 0\cdot485).$$

Vector $\mathbf{q}$ is a linear combination of $\mathbf{p}$ and $\mathbf{r}$, whilst RS = $\mathbf{s}$ is a linear combination of $\mathbf{p}$, $\mathbf{q}$ and $\mathbf{r}\times\mathbf{q}$. Solving $\mathbf{q} = \mathbf{p} + k\mathbf{r}$ for $k$ to make $\mathbf{q}\cdot\mathbf{r} = 0$, we obtain $k = -1\cdot059$. Therefore $\mathbf{q} = (0\cdot288,\ -0\cdot318,\ 0\cdot188)$ and $\mathbf{e_q} = (0\cdot615,\ -0\cdot678,\ 0\cdot402)$. The third mutually orthogonal unit vector at Q is $\mathbf{e_r}\times\mathbf{e_q} = (0\cdot621,\ 0\cdot104,\ -0\cdot777)$. The branch angle at Q is obtained from $\mathbf{e_p}\cdot\mathbf{e_r}$ and is $\theta = 47°$.

We are now in a position to define the next branch, RS. A vector of length $L$ at Q parallel to RS = $\mathbf{s}$ is

$$L\cos\theta\,\mathbf{e_r} + L\sin\theta\cos\phi\,\mathbf{e_q} + L\sin\theta\sin\phi\,\mathbf{e_{r\times q}}.$$

There is the special case noted above in which the new branch is in line with its parent, corresponding to $\theta = 0$; the new branch is then simply $L\mathbf{e_r}$. Suppose it is required to grow from $\phi = 50°$ azimuth at an angle of $\theta = 30°$ to QR, and have a length $L$ of 80% of QR, which is $0\cdot33$ length units. In the example this has components

$$(0{\cdot}138,\ 0{\cdot}209,\ 0{\cdot}139) + (0{\cdot}065,\ -0{\cdot}072,\ 0{\cdot}043) + (0{\cdot}078,\ 0{\cdot}013,\ -0{\cdot}098) = (0{\cdot}282,\ 0{\cdot}149,\ 0{\cdot}083)$$

and so the coordinates of point S are obtained by adding this to the position vector of R, giving S at $(1\cdot982,\ 0\cdot449,\ 2\cdot683)$ with respect to the base of the tree trunk, O.

An example is given in Figure 23 of the tree obtained by taking the 2D tree of the left of Figure 9 and turning each new branch out of the plane of the previous two branches by 40°. A BASIC procedure to carry out the above is listed in Figure 24. I have added a few lines of code to this to generate a third central branch at each node, the 3D version of Figure 9, right. With this a selectable apical dominance ratio can be introduced, whereby the central branch at each node has a different scale than the side branches. Two examples are shown in Figure 25, the left spreading and looking somewhat like an oak, the right growing vertically like a conifer.

The silhouettes can be made to look more like real trees if some randomness is introduced into all their parameters, including the number of branches per node. To this end I have introduced random numbers $N = \sqrt{\sqrt{n_1 n_2 n_3 n_4}} - 0\cdot5$ where each $n_j$ is randomly distributed over (0, 1). The distribution of $N$ spans $(-0\cdot5,\ +0\cdot5)$ and peaks around 0, giving something of a bell-shaped spread to its values which is more realistic than an unmodified

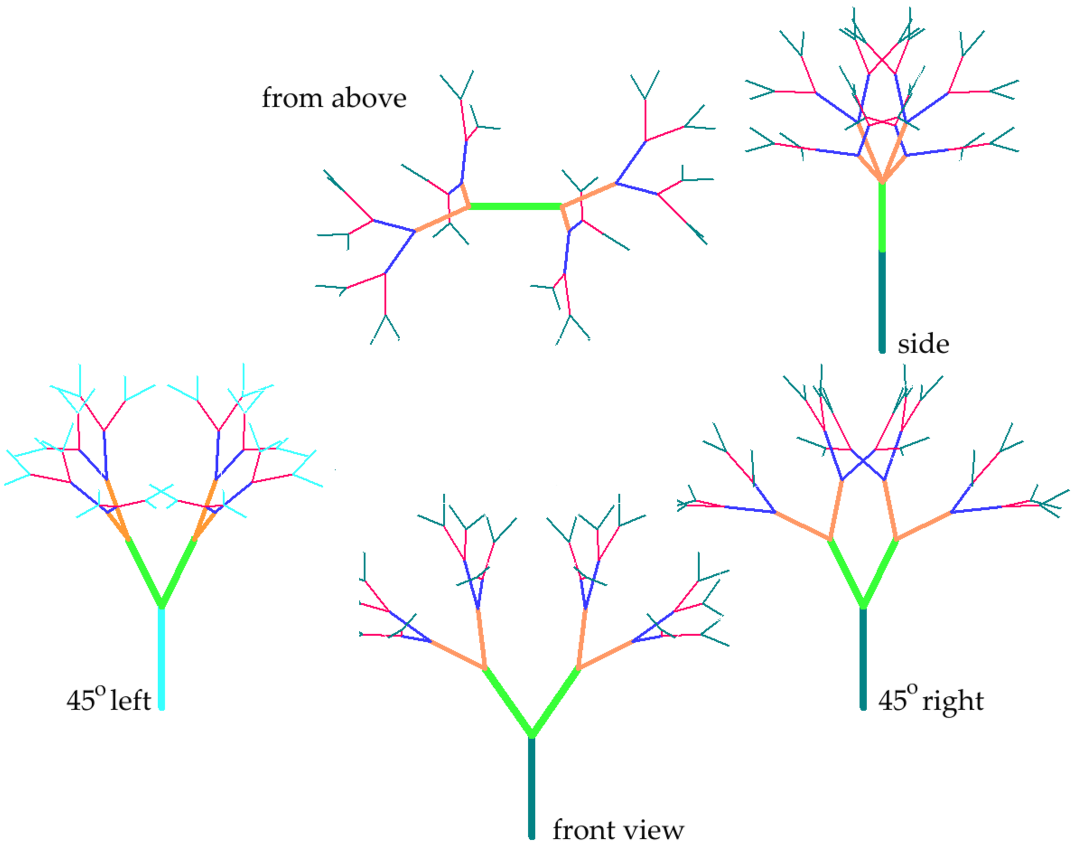


Figure 23: Five views of the same fractal tree, depth 5, obtained from Figure 9 by setting the branch polar angle to 35° and azimuths to 40° and 220° at every node. Each iteration is coloured differently.

uniform distribution. At the close of §3 I gave two examples of the program emulating a tall conifer and a spreading deciduous tree. Here in Figure 26 is a third example, similar to many young trees.

All the example so far have been of bare trees. Adding leaves to fractal trees is a matter of replacing the end twigs with a leaf-like element, preferably oriented as if towards the sun. The only elaboration I have mode is to draw the last line at the ends of twig with a short wide stroke. Three silhouette views of two trees in leaf are shown in Figure 27.

```
PRINT"Create a regular fractal tree with 2 new branches per node and stepped increments in azimuth."
length = 150         : REM Length of first branch in pixels.
polar = 35           : REM angle of new branch away from direction of parent.
azmth = 40           : REM angle of turn of new branch. i.e between planes PQ+QR and QR+RS.
scale = 0.8          : REM Length factor of new branch relative to parent
ThickFactor = 1.5    : REM Thickness of parent branch relative to daughter.
REM Set the origins in (U, V) screen co-ordinaes of four of the views: left, right, central, above
LOrgU = 500 : LOrgV = 50 :   ROrgU = 2200 : ROrgV = 50
COrgU = 1400 : COrgV = 150 : AOrgU = 1400 : AOrgV = 1000
Px = -100: Py = 0: Pz = -100      : REM Dummy first point P placed in tree roots
Qx = 0: Qy = 0: Qz = 0            : REM Base of trunk
Rx = 0: Ry = 0: Rz = length/scale : REM Top of trunk

Depth% = 8           : REM Depth of recursion
MODE 14 : GCOL 2     : REM Display parameter and graphics colour of trunk.

PROCNewBranch(Px,Py,Pz,  Qx,Qy,Qz,  Rx,Ry,Rz, length, polar, azmth, Depth%)
REM This creates one branch of Y shape from trunk.
PROCNewBranch(Px,Py,Pz,  Qx,Qy,Qz,  Rx,Ry,Rz, length, polar, azmth+180, Depth%)
REM This creates other branch of Y from trunk.
STOP : END

REM ****************

DEF PROCNewBranch(Px,Py,Pz,  Qx,Qy,Qz,  Rx,Ry,Rz, length, polar, azmth, dpth%)
REM P, Q, R are the node positions, not the intervening vectors.
LOCAL NPx, NPy, NPz,  NQx, NQy, NQz,  NRx, NRy, NRz
REM  IF dpth% = Depth% polar = 0 : azmth = 0

px = Qx-Px : py = Qy-Py:  pz = Qz-Pz  : REM Vector p = PQ
rx = Rx-Qx : ry = Ry-Qy:  rz = Rz-Qz  : REM Vector r = QR
p_sq = px^2 + py^2 + pz^2 : modp =SQR(p_sq)
unitpx = px/modp : unitpy = py/modp : unitpz = pz/modp
r_sq = rx^2 + ry^2 + rz^2 : modr = SQR(r_sq)
unitrx = rx/modr : unitry = ry/modr : unitrz = rz/modr
k = -FNdot(px,py,pz, rx,ry,rz)/r_sq
qx = px + k*rx :  qy = py + k*ry :  qz = pz + k*rz
q_sq = qx^2 + qy^2 + qz^2 : modq =SQR(q_sq)
unitqx = qx/modq : unitqy = qy/modq : unitqz = qz/modq
rXqx = qz*ry-qy*rz : rXqy = qx*rz-qz*rx : rXqz =qy*rx- qx*ry
rXq_sq = rXqx^2 + rXqy^2 + rXqz^2 : modrXq =SQR(rXq_sq)
unitrXqx = rXqx/modrXq : unitrXqy = rXqy/modrXq : unitrXqz = rXqz/modrXq
sx = length*(COSRAD(polar)*unitrx + SINRAD(polar)*(COSRAD(azmth)*unitqx + SINRAD(azmth)*unitrXqx))
sy = length*(COSRAD(polar)*unitry + SINRAD(polar)*(COSRAD(azmth)*unitqy + SINRAD(azmth)*unitrXqy))
sz = length*(COSRAD(polar)*unitrz + SINRAD(polar)*(COSRAD(azmth)*unitqz + SINRAD(azmth)*unitrXqz))
Sx = Rx + sx : Sy = Ry + sy : Sz = Rz + sz   : REM Position of tip of new branch.
NPx = Qx : NPy = Qy : NPz = Qz    : REM Updating co-ordinates ready for next iteration.
NQx = Rx : NQy = Ry : NQz = Rz
NRx = Sx : NRy = Sy : NRz = Sz

width%=INT(ThickFactor^dpth%+0.5)
VDU 23,23,width%;0;0;0;           : REM sets the thickness for drawing the lines.

REM Draw the trunk in three of the views.
IF dpth% = Depth% THEN
  GCOL 1
  MOVE LOrgU, LOrgV:  DRAW LOrgU, LOrgV+length/scale
  MOVE ROrgU, ROrgV:  DRAW ROrgU, ROrgV+length/scale
  MOVE COrgU, COrgV:  DRAW COrgU, COrgV+length/scale
ENDIF
REM Add the next branches
GCOL dpth%
MOVE Rx+LOrgU, Rz+LOrgV:  DRAW Sx+LOrgU, Sz+LOrgV
MOVE Ry+ROrgU, Rz+ROrgV:  DRAW Sy+ROrgU, Sz+ROrgV
MOVE Rx+AOrgU, Ry+AOrgV:  DRAW Sx+AOrgU, Sy+AOrgV
MOVE 0.7071*(Rx+Ry)+COrgU,  Rz+COrgV:  DRAW 0.7071*( Sx+Sy)+COrgU, Sz+COrgV : REM View from 45 degrees right.
REM MOVE 0.7071*(-Rx+Ry)+COrgU, Rz+COrgV:  DRAW 0.7071*(-Sx+Sy)+COrgU, Sz+COrgV : REM View from 45 degrees left.

IF dpth% > 0 THEN
  PROCNewBranch(NPx, NPy, NPz,  NQx, NQy, NQz,  NRx, NRy, NRz, length*scale, polar, 40, dpth%-1)
  REM Creates one branch of Y at node
  PROCNewBranch(NPx, NPy, NPz,  NQx, NQy, NQz,  NRx, NRy, NRz, length*scale, polar, 40+180, dpth%-1)
  REM Creates other branch of Y
ENDIF
ENDPROC

REM ^^^^^^^^^^^^^

DEF FNdot(ax,ay,az, bx,by,bz)
=ax*bx + ay*by + az*bz
```

Figure 24: BASIC code to generate regular fractal trees with step advances in azimuth, two branches per node. Figure 11 in §3 was generated with this code.

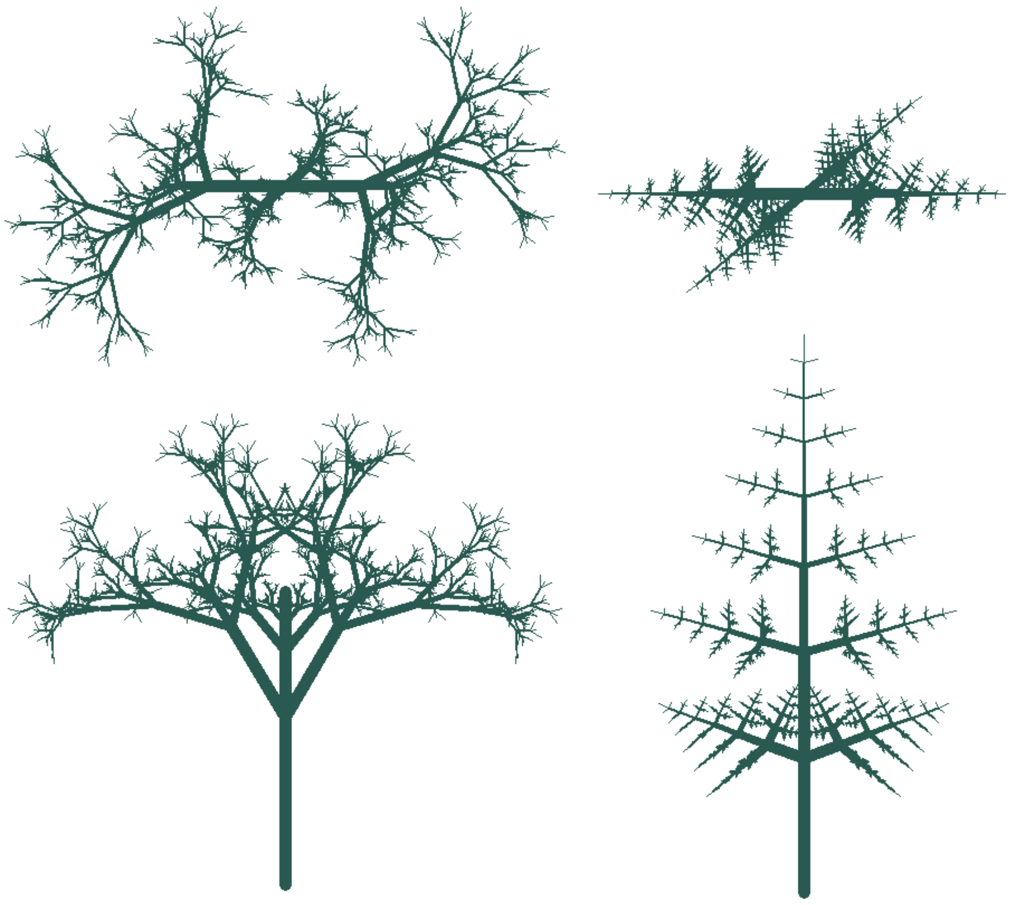


Figure 25: Two regular fractal trees with 3 branches per node drawn to recursion depth 6, azimuthal increment 40°. Left: apical ratio $0{\cdot}4/0{\cdot}6$, branch polar angle 40°. Right: apical ratio $0{\cdot}8/0{\cdot}4$, angle 75°. Views from 45° right and from above.

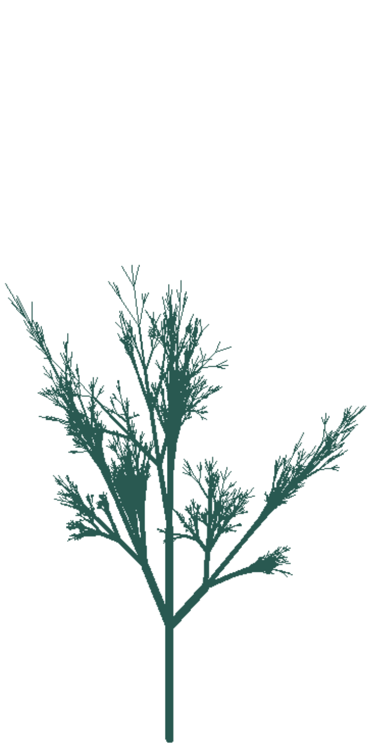


Figure 26: Three silhouette views 45° apart plus the view from above of a randomised fractal tree similar to many young real trees.

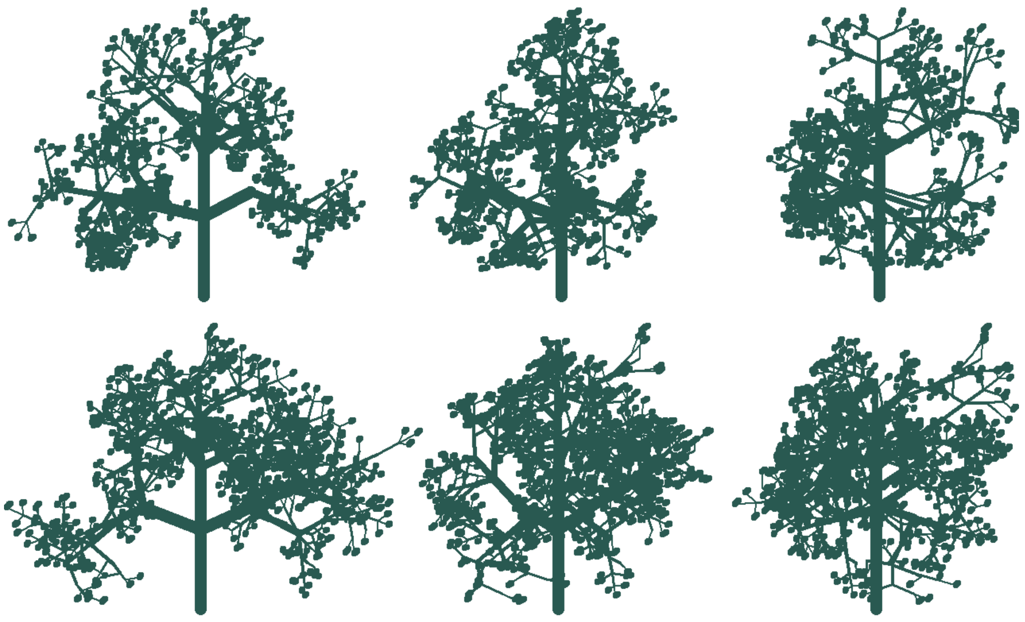


Figure 27: Three views each of two random fractal trees (above and below) with crude emulation of leaves. Some branches are randomly omitted to give the impression of an old tree which has lost branches.