

Language for definition of L systems

Formal definition of L systems and interpretation parameters are defined using a special purpose simplified language. For example, this a definition of an L system that arranges elements according to a cylindrical model of phyllotaxis.

```
Lsystem CylindricalPhyllotaxis;

define N = 100;           • number of elements
define h = 0.02;        • vertical distance between adjacent elements
define r0 = 8;          • radius at the base
define rf = 8;          • radius at the top
define di = (r0-rf)/N;  • variation of radius

colorspace = hsb;
step = 0.3;
angle = 1.0*fibangle;
baseelement = {
  shape = quad;
};

useralphabet = {A,B};
lifespan = N+1;

axiom = @(40,1,0.9)A(r0);
developmentrules = {
  p1: A(r) : true -> [^(90)f(sqrt(r))B]f(h)/A(r-di);
  p2: B    : true -> ^^(90)F;
};
```

Formal definition of L system must be given but interpretation parameters are optional. Default values are used whenever they're omitted. Each definition is ended with a semicolon that acts as a separator of definitions. Sets are bounded with curly brackets and their elements are separated with commas. Definitions may span more than one line and can be place in any order. Identifiers and symbols are case sensitive but not reserved words.

L system name

Each L system is assigned a title using the reserved word LSystem as in

```
Lsystem Hilbert3d;
```

It is not necessarily the same identifier as the file where the definition is stored.

Notes

Notes can be added to the definition. All characters after a midpoint (‘.’) to the end of the same line are ignored. For example

```
define w0 = 1/sqrt(2);    • initial width
```

Constant parameters

Constant parameters can be defined to make an easier reading of expressions and emphasize relevant values, as in

```
define N = 100;           • number of elements
define h = 0.02;        • vertical distance between adjacent elements
define r0 = 8;          • radius at the base
```

```
define rf = 8;           • radius at the top
define di = (r0-rf)/N;  • variation of radius
```

User defined symbols

Every L system alphabet includes a minimum subset of predefined symbols needed for usual operations: rotations, branching, etc. They don't need to be declared. User defined symbols are ignored by the interpreter of modules. They're added to the alphabet as in

```
useralphabet = {A, B, w, s};
```

Every symbol appearing in the axiom and production rules must belong to the alphabet. ASCII characters can be used as symbols except those already predefined

```
F, f, [, ], %, -, +, |, &, ^, /, \, #, !, @, ", ' ,
```

and those that might cause syntactical ambiguities

```
;, :, <, >, (, ), ,
```

Axiom

Axiom is set with

```
axiom = FA(1);      • start from internode
```

Note that the semicolon is not a symbol but the separator of definitions. Axiom cannot be an empty string of modules.

Production rules

There must be at least one production rule. Each rule consists of: identifier, predecessor module, condition and one or more successors (for stochastic rules).

```
developmentrules = {
  p1 : A(v) : v<16 -> [-FB(v)][+FB(v)]FA(v+1);
  p2 : B(v) : v>0 -> FB(v-1);
};
```

Note that the semicolon is not a symbol but the separator of definitions. Rules can span more than one line. When more than one rule match a module the first defined is selected. Unconditional rules can be written as

```
p1 : A : true -> [-FB][+FB]FA;
```

Stochastic rules have more than one successor. Each successor has an associated constant probability factor. For proper working they must sum one.

```
p20 : V(c): c<>0 -> {
  0.34 : V(c),
  0.33 : B(0),
  0.33 : B(1)
};
```

Derivation steps

The number of derivation steps is set using the reserved word *lifespan*.

```
lifespan = 8;
```

Expressions on the right are rounded to the closest integer.

Environment parameters

Environment sensitivity is supplied via a set of environment parameters. They can be accessed from any module in production time. Their use is simpler than that described in environment sensitive L systems. The available parameters and their meanings are shown in the following table.

	Px, Py, Pz	Hx, Hy, Hz	Lx, Ly, Lz	Ux, Uy, Uz	Nr
Meaning	Position (P)	Head direction (H)	Left direction (L)	Up direction (U)	Level of branching

According to the following rule, its condition will be true only when the current module position is within a circle centered in the origin and radius r . If it applies color number will be set to current depth recursion so branches at the same order of branching will be drawn with a distinct color.

```
p1 : A : sqrt(sqr(Px)+sqr(Py))<r → #(Nr)+(10)FB;
```

Random generators

Unless specified, random seeds are chosen randomly. However they can be set to a given value as in

```
randomseed = 850;
```

to achieve the same series of random values multiple times.

Colors

Colors can be specified using color tables or color spaces. Color tables are selected giving their corresponding file name. For example:

```
colortable = "slime.tcl";
```

Colors can also be dynamically defined according to relevant parameters of the L system. For example, setting color randomly or as a function of height. They can be defined using RGB or HSB color spaces. Usually HSB is more suitable.

```
colorspace = HSB;
```

All color components must lie in the range $[0, 1]$ except hue that is specified as a $[0, 360]$ degree. Values outside the given range are saturated.

Base elements

Line is the default shape used for drawing, but other different shapes can also be employed: quads, crossed-quads, boxes, cylinders and tubes. Additional features can be set for each shape. Use the *baseelement* reserved word for this purpose.

Cylinders and boxes can have caps open or closed. For cylinders, the number of sides must be between 3 and 64.

Next are some examples:

```
baseelement = {
  shape = cross;
};

baseelement = {
  shape = box;
  caps = closed;
};

baseelement = {
  shape = cylinder;
  caps = open;
  sides = 18;
};
```

Tubes are built by sweeping a given flat cross-section along an axis. The points that define the axis are defined by the drawing modules. During interpretation, a Hermite curve interpolates those points and a contour is swept along it creating a tubular shape.

Four additional features are used to adjust the tube appearance. The cross-section is specified by its corresponding file name. If no cross-section is given, only the axis will be created. The smoothness of interpolation is given by the number of strips between adjacent points in the axis. Roundness of the interpolated strips is adjusted by a tension factor. The tension value that produces more 'natural' forms is 1.2. Finally, another parameter selects the way end points of the axis are interpolated.

```
baseelement = {
  shape = tube;
  smooth = 4;
  tension = 1.2;
  crosssection = "star 7p.con";
  endpoints = central;
};
```

Operators and functions

The usual operators can be used to construct expressions: +, -, *, /, <, >, =, <=, >=, <>, *and*, *or*, *not*, *xor*. In addition, a set of mathematical functions are available: *abs(x)*, *arccos(x)*, *arcsin(x)*, *arctan(x)*, *arg(x)*, *ceil(x)*, *cos(x)*, *deg(x)*, *div(x,y)*, *exp(x)*, *floor(x)*, *frac(x)*, *int(x)*, *ln(x)*, *log10(x)*, *log2(x)*, *max(x,y)*, *min(x,y)*, *mod(x,y)*, *norm(x,y,z)*, *power(x,y)*, *rad(x)*, *randomn(x,y)*, *randomu(x,y)*, *round(x)*, *sign(x)*, *sin(x)*, *sqr(x)*, *sqrt(x)* and *tan(x)*.

Constants

Some useful constants are defined by default: *false*, *true*, *pi*, *fibangle* and *goldratio*.