



## L-System Design

Creating an L-System for a particular purpose is neither easy nor intuitive, but, when successful, the results can be more than worth the effort. And as is the case with many such things, the process becomes easier with practice and experience.

When designing L-Systems, it is important to keep in mind their basic properties and their inherent problems.

### The Fractal Nature of L-Systems

L-Systems fundamentally are fractal generators. Although it is possible to design L-Systems that produce simple, easily understood patterns, the L-System mechanism by its nature is fractal.

This fractal nature comes from three sources:

- parallel operation on characters (global uniformity)
- characters defined in terms of themselves (recursion)
- repeated application of the rules (iteration)

The way characters can be defined in terms of themselves is of central importance. A simple example is

```
seed:   A
rules:  A → ABA
        B → BBA
```

Here both **A** and **B** are defined in terms of themselves and each other. Repeated applications of the rules produces increasingly long and intricate combinations of the two characters:

```
A
ABA
ABABBAABA
ABABBAABABBABBAABAABABBAABA
```

...

Although the rules are simple, the patterns that develop are nonetheless complex and not easy to characterize.

### The Seed

The seed, with which generation begins, is not particularly important. In

most L-Systems, the seed is a single character. The seed can be a string of characters, but an L-System with such a seed can always be replaced by an L-System whose seed is a single character.

Consider this example:

```
seed:   ABCBA
rules:  A → BC
        B → AB
        C → CB
```

A new character can be added as the seed and a new rule can be added replacing it by the original seed:

```
seed:   D
rules:  D → ABCBA
        A → BC
        B → AB
        C → CB
```

The only difference between these two L-Systems is an additional generation in the second. Note that D only appears once.

## Alphabet

The alphabet of the characters used in an L-System really only matters as to the number of characters. Characters are arbitrary. They may be chosen for mnemonic value, but until the interpretation of a string generated by an L-System, they have no meaning.

For example,

```
seed:   A
rules:  A → ABA
        B → BBA
```

and

```
seed:   3
rules:  3 → 3X3
        X → XX3
```

are equivalent.

## Generation Length

An inherent property of L-Systems is increase in length of successive generations. In fact, this limited early work on L-Systems at a time when the amount computer memory was very available was very small.

It is possible to design L-Systems in which generation length does not increase. An example is

```
seed:  A
rules: A → B
       B → A
```

which generates

```
A
B
A
B
...
```

Such L-Systems are both contrived and trivial.

When a rule specifies replacement by more than one character, generation length increases. This problem is addressed in a subsequent section.

### Character Relationships

The way that characters are defined in terms of themselves and each other has many effects on L-System generation.

If not all characters appear in all rules, there many be successive generations that have essentially different characteristics.

A simple and trivial example is

```
seed:  A
rules: A → BB
       B → CC
       C → AA
```

for which the generations are

```
A
BB
CCCC
AAAAAAA
```

BBBBBBBBBBBBBBBBBB

...

A subsequent section addresses the issue of character interaction in more detail.

### What is Possible

L-Systems are only one of many kinds of formal grammars [11]. Different kinds of grammars have different “expressive power”. The issue of expressive power is of both theoretical and practical importance.

Expressive power, roughly speaking, is a measure of what kinds of patterns a formal grammar can produce. Expressive power is measured more in terms of what patterns can be excluded than what may be included.

For example, almost all kinds of formal grammars can produce palindromes, but they inevitably produce other patterns as well. That is, non-palindromes cannot be excluded by grammars of most kinds. L-Systems can generate purely palindromic sentences and in this sense are more powerful than most other kinds of formal languages.

### Interpretation

Although interpretation falls outside the scope of L-Systems proper, it is a power design tool and its possible use needs to be kept in mind when L-Systems are designed.

An L-System intended to produce a profile draft may not require any interpretation other than the think of the characters as blocks. On the other hand, an L-System designed to draw a pattern may require interpretation of characters as navigation and drawing actions [1].

But interpretation can be used to change L-System strings in arbitrary ways, including reordering them, deleting characters, and so forth. In some sense, there is no limit to the power of interpretation.