Lindenmayer Tool

A Lindenmayer-System Model Exporter

A better name probably exists.

Michael Gircys

mg12vp - 5269550

COSC 3P98 - Computer Graphics

Prof. Brian Ross

2014-01-08

Basic UI Stuff

The program contains a preview window a simple menu with a minimal set of commands.

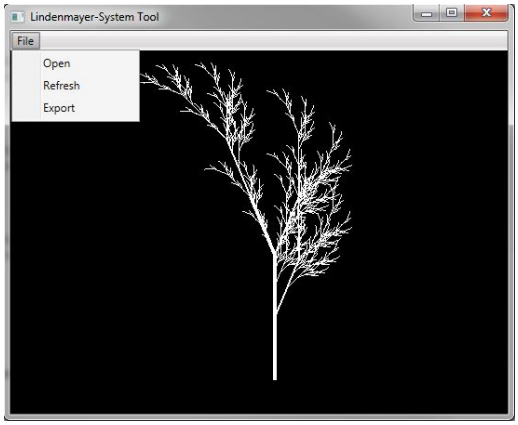

Figure 1 - The main program display

The Open command presents a Open File dialog, and will then load and generate an L-System from the a selected .lsys file

The Refresh command will recompute the loaded L-System. This may be useful for stochastic systems which provide some randomness in replacement rules.

The Export command presents a Save File dialog, and will then attempt to export the computed L-System geometry into a Wavefront Object formatted file.

The preview window is adjustable with mouse operation: zooming in with the scrollwheel, translating by holding left click while moving the mouse, and returning to default with middle click. The preview window will resize with the window.

Basic L-System Stuff

Lindenmayer Systems encompass a number of rewriting systems, the abilities of which can simultaneously fall across a number of different levels of the Chomsky classes of grammars (Lindenmayer 3). The key principle is that a grammar makes replacements simultaneously (as opposed to consecutively) to create organic fractals.

While L-Systems have been around for some time now, a detailed discussion and explanation can easily extend far beyond the limits of this document. For more details, see *The Algorithmic Beauty of Plants* referenced later, or <http://en.wikipedia.org/wiki/L-system> (A .pdf of TABOP is linked here, but appears to be inaccessible at the moment).

Implementation Details

L-Systems:

Initially, a simplistic approach to token parsing was used, treating each character of the string as a token. To allow for parametric systems, a list of regular expression matches were used to allow for optional parametric expressions. Replacements are made by matching a token to a rule's stored LHS value, and placing one of the rules RHS values into the new L-System string.

[Some difficulty was foreseen in allowing parametric expressions, as while simple evaluations and replacements could be made, proper support would require complex scripting evaluation; variables and constants would need to be evaluated with something like LUA, and the evaluator would need to be able to handle multiple inline inequalities (ie. "1<d<5,e=2"). Complete parametric system functionality was sacrificed for proper export functionality.]

An LSystem object holds all configuration information for a given system, and has members to allow for loading .lsys files, performing replacement iterations, and rendering L-System interpretations. The LSystem object is then responsible for managing all branching commands while interpreting the produced string, and passes all other commands to the topmost LTurtle object in the stack.

LTurtle objects track position, orientation, width, and other expected values while iterating commands in the interpreted produced string.

Export:

Given a stored list of geometries - at current these are only stored line objects - each line is handled on its own. The required vertices are serialized and sent to output, with the absolute index of the vertices recorded. This is followed by faces being constructed which make reference to the newly output vertex indices, and are subsequently output. The Wavefront Object specification gives no obvious regulation for the positioning or divisibility of the vertex data, so it is possible to have the faces immediately preceded by the vertices it references.

An optimization I put into the export class was to first search through the list of lines and provide an optimized list, merging adjacent lines of the same width. This managed to reduce the number of faces by about half in my tests. The reduction wasn't quite as much as I had hoped for, but considering the exponential branching at the ends, it was a reasonable improvement.

The export function uses a simplistic method to draw lines. Each stored line object is represented as a square with a normal facing +Z. Ideally, the number of the faces circumscribing the line would be configurable, though the high number of vertices and faces with this simple geometry is cause for concern. An idea I had considered post facto may be to relate the number of faces per segment to the segment's width, or perhaps its optimized length.

L-System File Format

The program loads a custom L-System specification file (.lsys) based on the format provided for examples in *The Algorithmic Beauty of Plants*.

Default values for a new L-System are as follows:

```
Iterations                = 10;
Step Size                 = 5.0
Rotation Angle            = 90.0
Initial Segment Width     = 1.0
Segment Width Decrement   = 0.1
Colour Palette            = #FFFFFF, #FF0000, #00FF00, #0000FF
Seed / Initial String     = "S"
```

Options

Comments can be included by beginning the line with a "#".

L-System settings can be configured as shown below, replacing `n`, `f`, and `h` for valid integers, floats, and hex values respectively.

The number of replacements to perform can be set with

```
Iterations : n
```

The length of each segment (forward) can be set with

```
StepSize : f
```

The angle to rotate for each rotate command can be set with

```
AngleIncrement : f
```

The initial width of segments can be set with

```
WidthStart : f
```

The proportion of the segment to be decreased with "!" can be set with

```
WidthDecrement : f
```

(0.10 reduces width by a compound 10% each decrement)

A colour palette can be defined as follows

```
Colours : h
Colours : h, h, ...
```

The first call will clear the default colour palette before adding the provided colours, and additional calls with append the palette. One or more colour can be defined provided they are separated with commas. The colour format is a 6-digit hex value in RGB format. The values should not have any preceding markers.

Each of the above statements can be provided any number of times with the last provided value being used, with the noted exception of colour palette.

Rules

A seed, or initial string, can be loaded with

```
w : S
```

Where 'w' is a constant tag signifying the initial string, and 'S' is some expression.

The program can support loading rules for DOL-systems, as well as stochastic and parametric systems.

The main format of the rules to be provided is as follows (optional components are in square brackets):

```
<tag> : <LHS> [ : <condition> ] [ (<probability>) ] --> <RHS>

<tag>         - An identifier for the rule; for human use and seed definition.
                Seeds must have tag "w", and rules must be in form "pn", with
                integer n
<LHS>         - The left side of a production rule; a variable to match.
                There must be exactly one character outside of any bracketed
                parameters.
<RHS>         - The right side of a production rule; the replacement.
<condition>   - constraint expressions on parameterized variables
<probability> - a weight (float) assigned to the rule in the case of stochastic
                systems with multiple replacements for the same match. Default: 1.
```

Stochastic rules can be constructed by providing multiple rules with identical LHS expressions. Tag expressions may be different. Each rule will have a default probability weight of 1 unless otherwise specified.

Currently, the program supports loading of rules for parametric systems, but provides only partial interpretive functionality. Conditions are not checked, and parameters are not used or replaced. Simplified parametric operations can be supported by treating parameterized variables with unique parameterized expressions as separate variables.

Expressions are strings of L-System turtle symbols and other (terminal or non-terminal) characters. Currently, each character that is not a turtle symbol may be followed by an expression enclosed by parenthesis, though a rule must match the character and following expression exactly.

A number of examples have been provided in the folder "../Examples" relative to this document.

I quite enjoy *StochasticTestColours.lsys*, or *3DExample.lsys*, which was based on it. Have fun.

L-System Turtle Symbol Interpretation

(From *The Algorithmic Beauty of Plants*)

F	Move forward and draw a line.
f	Move forward without drawing a line.
+	Turn left.
-	Turn Right.
^	Pitch up.
&	Pitch down.
\	Roll left.
/	Roll right.
|	Turn around.
$	Rotate the turtle to vertical.
[	Start a branch.
]	Complete a branch.
!	Decrement the diameter of segments.
'	Increment the current colour index.
%	Cut off the remainder of the branch.

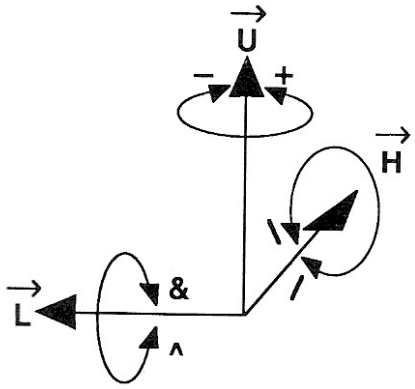

Figure 2 - Rotation Operations

Currently Unsupported Symbols:

{	Start a polygon.
}	Complete a polygon.
G	Move forward and draw a line. Do not record vertex.
.	Record a vertex in the current polygon.
~	Incorporate a predefined surface.

References:

Lindenmayer, Astrid, et al. *The Algorithmic Beauty of Plants*. New York: Springer, 1990. Print.

Wavefront Technologies. *Object File Specification*. Accessed 2013-12-18. <http://www.martinreddy.net/gfx/3d/OBJ.spec>

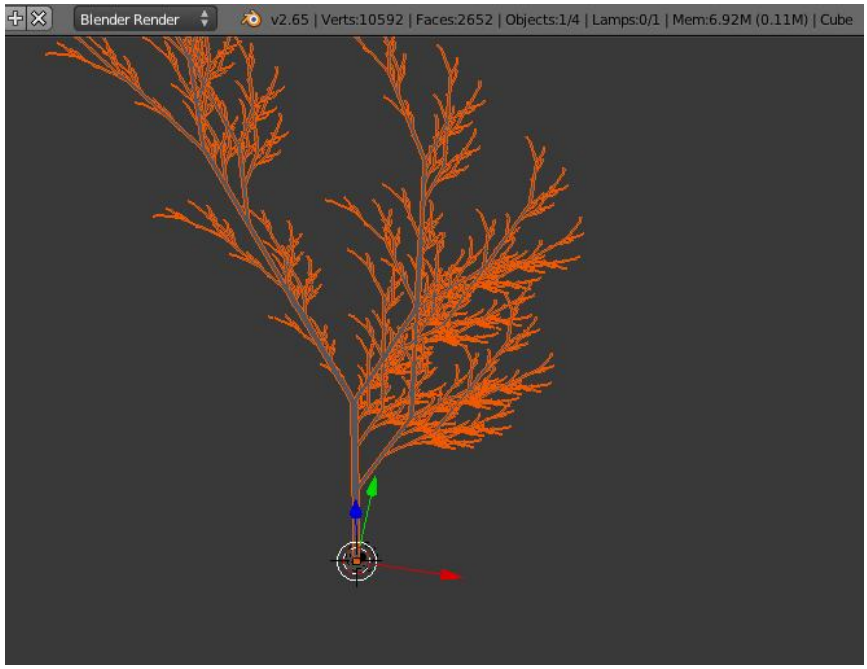

Figure 3 - The export actually worked? Wow.