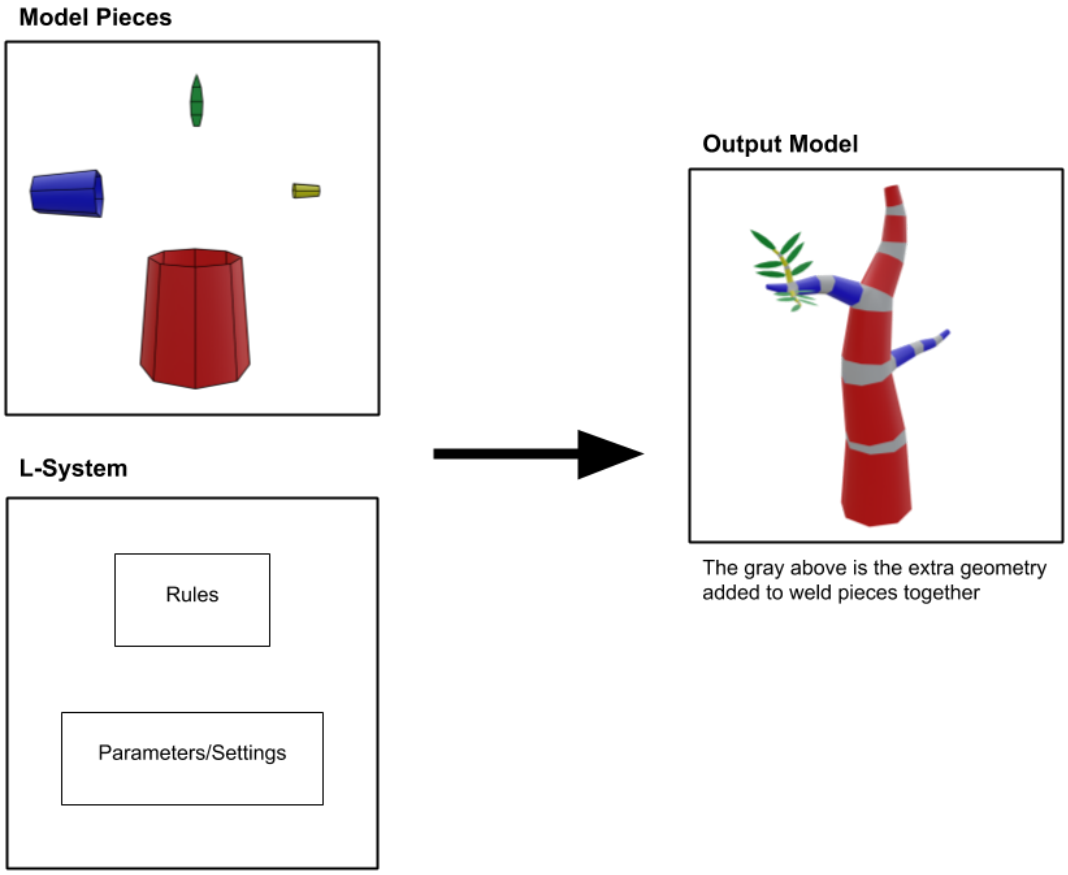
Jeremy Berchtold

September 30, 2020

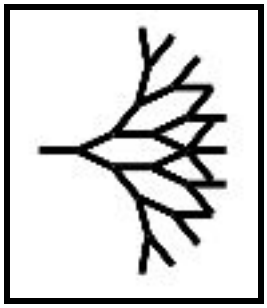
POMegranate: A Procedural Organic Modeling Tool



Overview

The goal of this project is to create a command-line tool to help users generate procedural organic models based on small pieces of base geometry. The organic growth will be modeled with user-specified L-systems. The output of the L-system will then be used to instance the user given base geometry pieces. From there, this tool will weld those pieces together to form one final model.

L-Systems



Base: `F`

Rules: `F=F[-F][+F]`

Angle: `25°`

Generations: `4`

L-systems can generate complex organic structures using simple rules. L-systems contain a starting base string, and then repeatedly apply rules across generations to generate structures.

Syntax

This project will primarily use the L-system syntax from Houdini's documentation. However, I need some way of differentiating between different base geometry types. I have two possible methods for this.

First, I may introduce new commands to link pieces of the L-system with base geometry types (e.g. trunk, branch, leaves). These new commands will associate `F` commands with base geometry types so they can be instanced to create the final geometry.

Second, I may keep the L-system commands unchanged, and instead add a filter on the generation or distance to origin (measured in the number of `F` commands used). The filter will allow the base of the L-system to be the trunk, slightly further to be large branches, slightly shorter to be small branches, and the furthest away to be leaves and fruit. I have not yet decided which of these two approaches to use.

Randomness

`~(a)`

Pitch / Roll / Turn random amount up to *a* degrees. Default 180.

Houdini's syntax has built-in ways to generate randomness, but I may expand it to include a mean and standard deviation for more control over the result.

User Input

The main input file will be a JSON or XML file. It will contain the following:

- L-system data (base, rules, etc.)
- List of named base geometry types (e.g. trunk, leaf, root, etc.)
- Paths for base geometry files (likely OBJ format)

sample.json

```
{
      "L-system": {
            "base": "F",
            "rules": [
                  "F=F[-F][+F]"
            ],
            "angle": 25,
            "generations": 4
      },
      "geometry": {
            "trunk": [
                  "trunk1.obj",
                  "trunk2.obj"
            ],
            "leaf": ["leaf.obj"],
            "branch": ["branch.obj"],
            "root": ["root.obj"]
      }
}
```

L-System to Geometry

Instancing

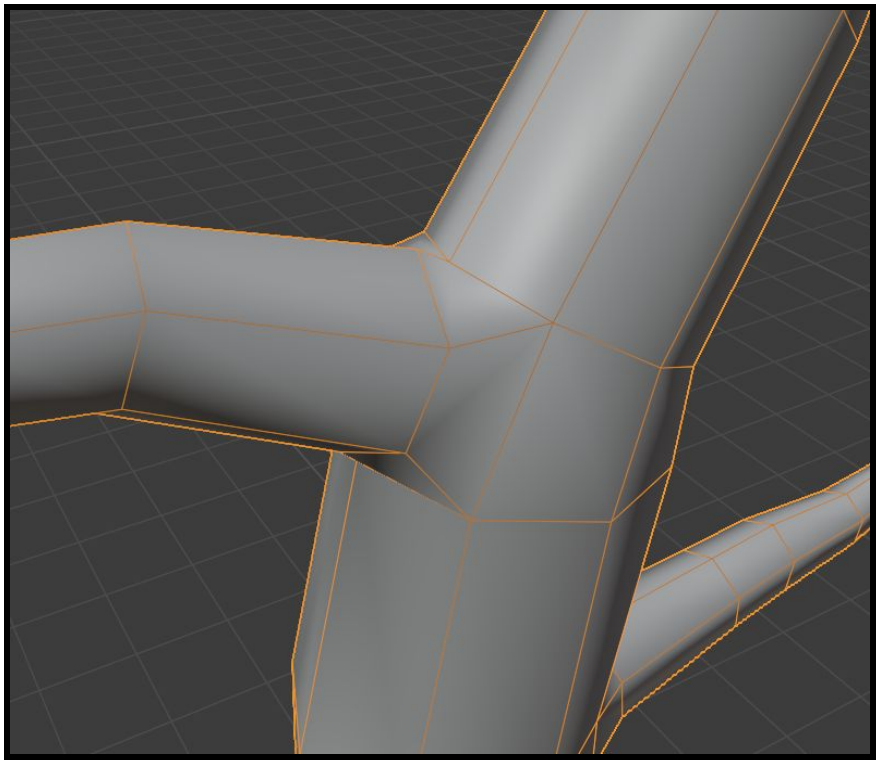
The base geometry for each type (e.g. trunk, branch) will be copied to all corresponding locations in the L-system (see cover photo). If there are multiple base models for a type, as seen with trunks in the file above, a random one will be chosen for each L-system location.

Welding

This will likely be one of the most difficult parts of this project. One the disconnected pieces of base geometry are instanced, they need to be connected, ideally with good topology. The only exception being leaves and fruit, as those may be instanced as disconnected geometry similar to a particle system on the surface.

In order to do this effectively without user help, I will likely have to restrict input geometry to match certain criteria. For example, all pieces (except leaves) need to have an edge loop at the top and bottom with the same number of vertices on each. Joining will then simply be bridging or vertex-merging those edge loops.

When the L-system goes in two or more different directions, figuring out the topology procedurally will be more difficult. In the example (see cover image), I was able to branch off of two nearby edge loops by restricting the branches to only be 6-vertex edge loops (shown below). The welding of the geometry together is the part of this project I am most unclear of and need to experiment to find a good solution.



Texturing

Texturing for all base geometry instances will be the same. All base geometry instances of the same type will share UV space. For example, all trunks will have the same texture. Since the UV space will be reused, most of the model will be texturable without much extra effort. For variation, users can include multiple uniquely textured OBJ files for a single geometry type (see trunks in sample.json above).

The only difficulty is the welded pieces (gray in the cover image). These have no texture. I am not sure yet what I want to do with these since I haven't decided how I will generate the welds yet. Once I have solved the welding problem, I can come back and solve this issue.

Resources

L-Systems

Houdini L-Systems

Houdini: Trees and Plants with L-Systems (note the intersecting topology where the tree splits, my project will attempt to fix this)

Randomness

3D noise functions. These will likely not be used as most L-system variables will be scalars (length, angle, etc.) and will be sampled uniformly or on a normal distribution.

Perlin noise

Worley noise

SpeedTree

SpeedTree is a tool for creating trees quickly and contains procedural elements, but it is not L-system based. I am including it as a resource to determine the components of a good virtual tree.

SpeedTree

SpeedTree Tutorial