Documentation

Peter Wildfeuer
OPT Industries: Design Task

peterwildfeuerdesign.com

Content

1. Sketch

A quick breakdown of the sketch and how it was interpreted.

2. Algorithm

A review of the algorithm and what changing parameters create.

3. Next Steps

What steps need to be taken to improve performance , add flexibility and facilitate performance applications.

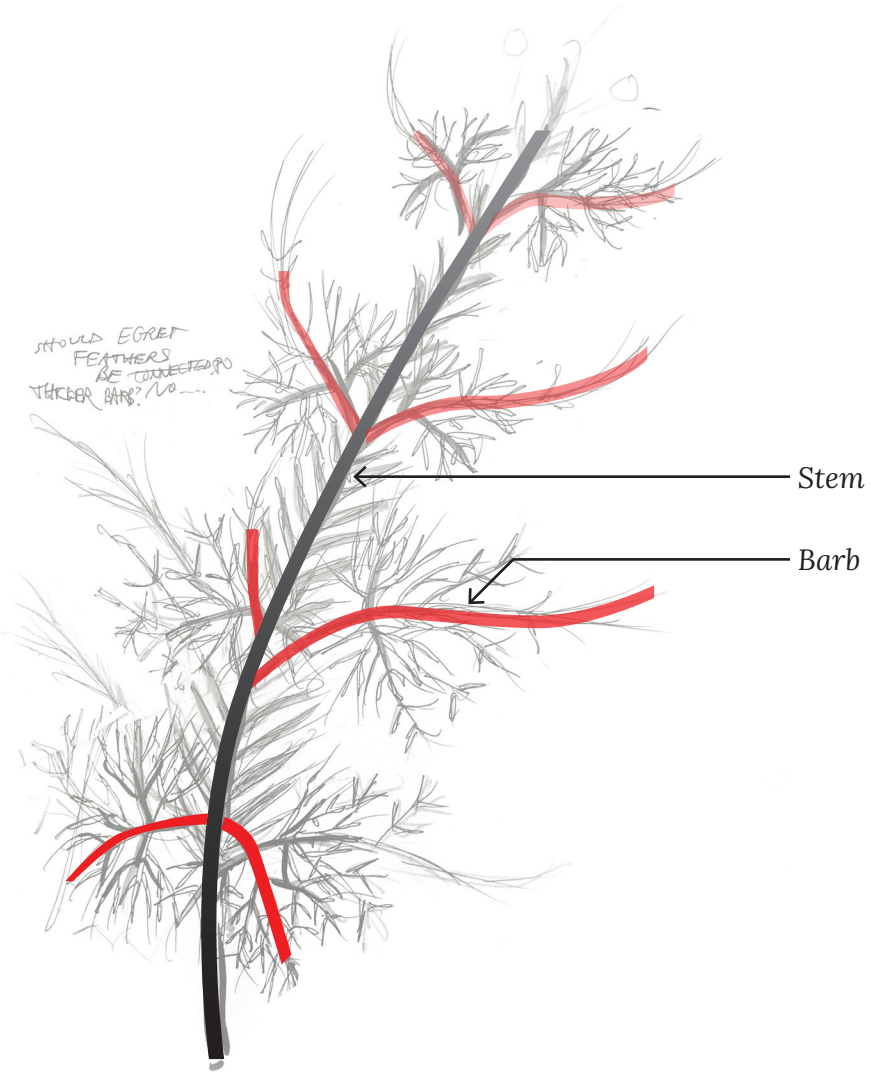
Sketch

Analysis

The sketch resembles a branching algorithm. Therefore by looking at each element in relationship to its previous members we can gain insight on how to properly parameterize this.

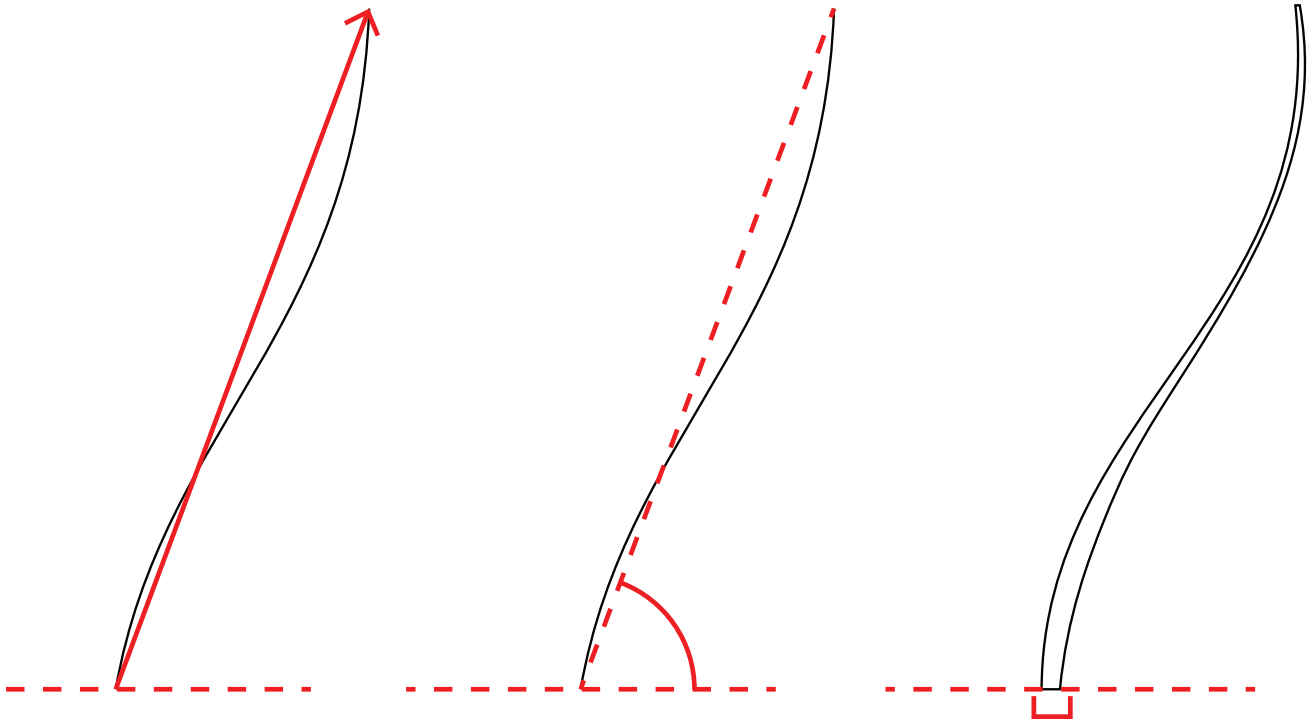
There doesn't seem to be a desire to focus on any singular item, rather than the overall form itself.

Finally, many traits seem to change with length down the "stem," including *the thickness of the lines, the length of the barbs and bushiness.*



Fading barb thickness based on Stem

Stem-Barb Traits



1 - *Barb Length*

The barb length is thought of as a fraction of the stem length ***(before entropy)***

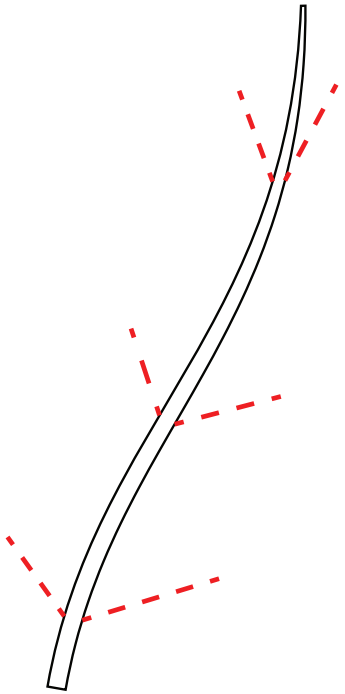
2 - *Angle*

The starting angle of the barb from the stem ***(before entropy)***

3 - *Radius*

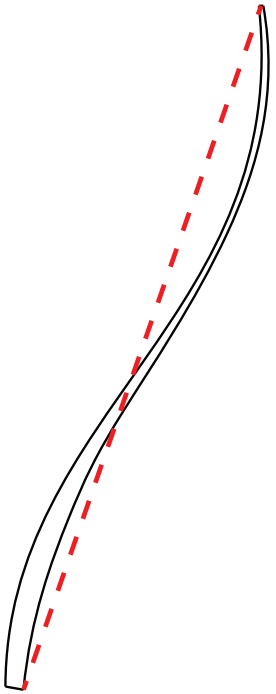
The initial radius of the barb is determined by its location on the stem.

Barb Traits



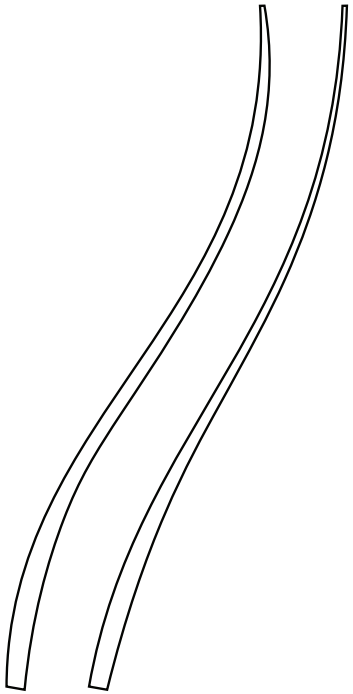
1 - *Barb Density*

The number of barbs to sprout from this one during the next pass



2 - *Curl*

The current barbs dedication to moving organically due to "entropy"



3 - *Thinning*

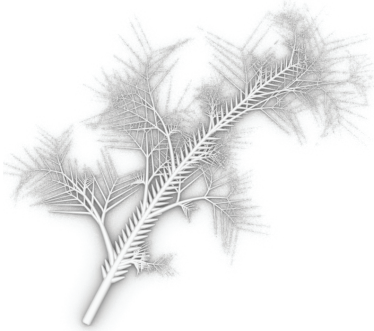
How quickly the cross sectional area reduces as we move down the barb path.

Algorithm

Controlling Generations

As a branching script it is heavily dependent on the lower generations for processing speed, mass and more, but the higher generations are vital to getting the overall expression right.

High Entropy



High Density

@ Higher generation

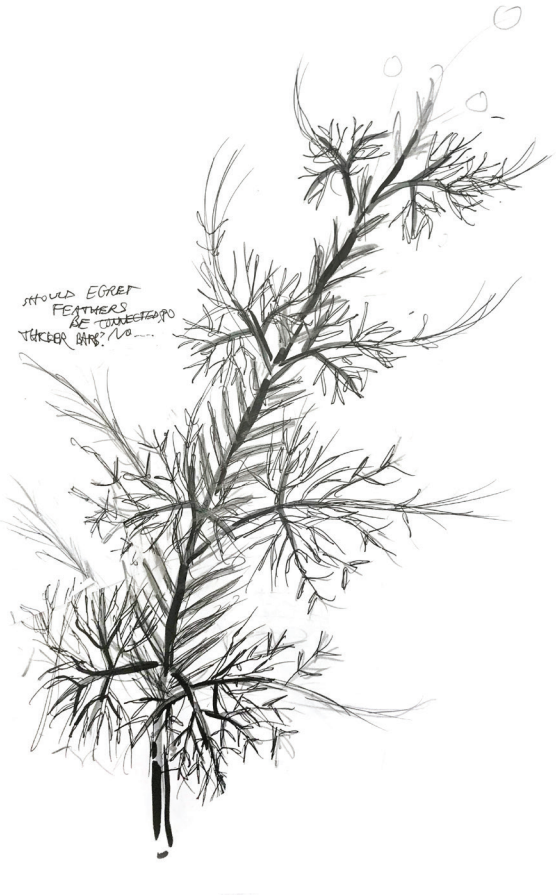
@ Lower generation

Parameters

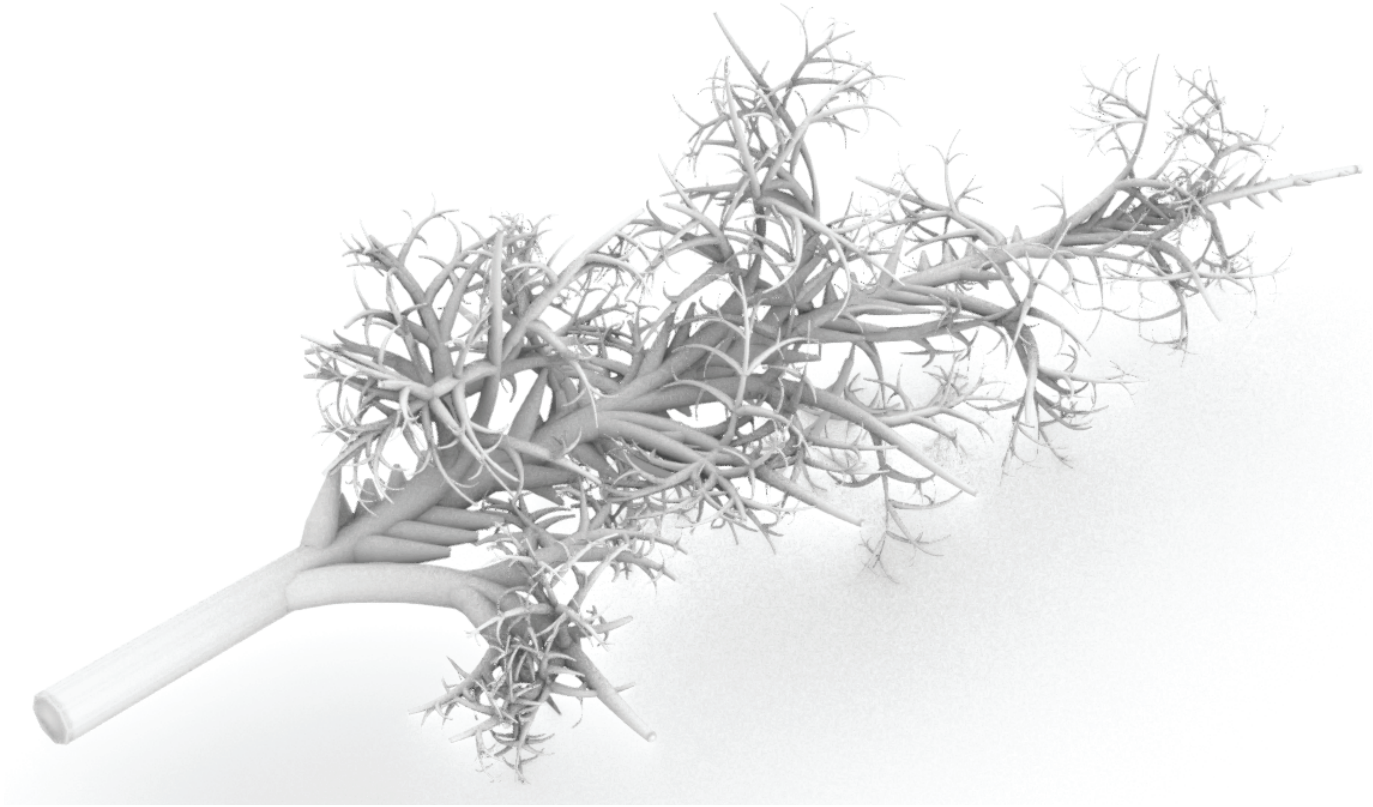
Below is a quick matrix of the parameters. ***There are 28 variables or 7 variables for each of 4 generations***

```
371     #### HYPER-PARAMETERS ####
372
373     # each list represents a different trait
374
375     angles =        [90, 90, 90, 90]
376     long_ratio =    [.2, .4, .5, .3]
377     barb_density = [.04, .03, .06, .06]
378     entropy =       [.2, .3, .5, .3]
379     curl =          [.8, .2, .3, .1]
380     thinning =      [1, .75, .5, .5]
381     short_ratio =   [.05, 0, 0, 0]
382
383     #
384
385     ######################
386
```

Sample Output



Sample 3D



Next Steps

Resuming Generations

The current script rebuilds the feather completely during every run, which is troublesome if you want to just change the top one or two generation of barbs.

We can write out the path information for each generation separately, and then import them to resume “growing” with different variables.

That way we can keep what we like about lower generations and change what we want with the higher ones.

Interlocking Barbs

Feathers can generally provide wonderful material properties, including wicking action, insulation and hydrophobicity.

All of these benefits come from the barbs of the barbs (or barbules) working together, as opposed to independently.

If we can create a system where barbules interlock between barbs we may be able to duplicate that effect.