Ari Danesh
ari@sidefx.com

Trees created in LSystems by Andy Boyd

WHAT ARE LSYSTEMS GOOD FOR?

LSYSTEMS CAN

- Create mid-field and far field trees and plants
- Form the geometry from which to create a hero tree
- Animate the growth of trees/plants/ molecular organisms
- Create Geometric Shapes

OBJECTIVES

- Introduction to LSystems
- By end of workshop feel comfortable reading Rules
- Be able to set up your own LSystems
- Understand how to decompose “your” tree into Sub-Systems
- Be able to create leafs and fruits

WORKSHOP WILL NOT COVER

- Texturing and Rendering Trees
- Shader Systems
- Animating Flowering
- Cacti and Succulents
- Pre-Modeled Organs

WHAT ARE LSYSTEMS

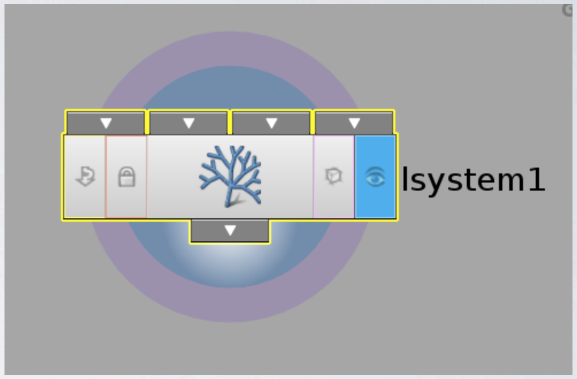
L-systems (Lindenmayer-systems, named after Aristid Lindenmayer, 1925-1989), allow definition of complex shapes through the use of iteration. They use a mathematical language in which an initial string of characters is matched against rules which are evaluated repeatedly, and the results are used to generate geometry. The result of each evaluation becomes the basis for the next iteration of geometry, giving the illusion of growth.

WOW, THAT WAS A MOUTHFUL

Think of LSystems as Genetic Sequences - A bunch of symbols are strung together to create trees, plants, and geometric shapes.

The strings call each other and replace simple instructions with more complex instructions. They evolve from a simple young tree to complex mature tree.

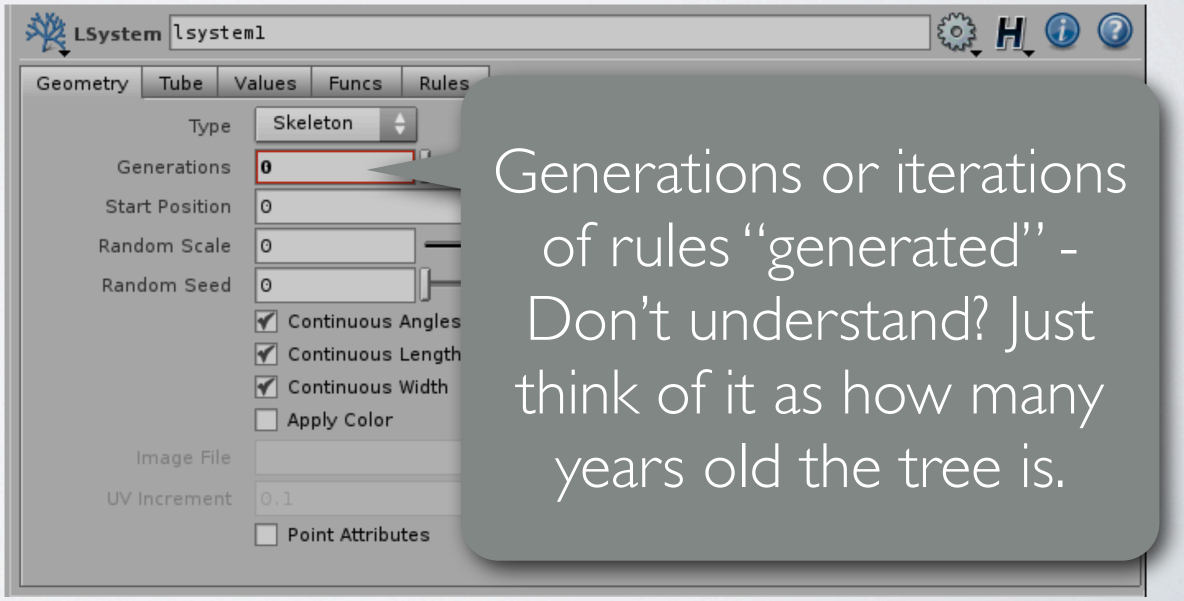
NODE HAS 4 INPUTS



- Drop down an "LSystem" node.
- We will discuss in detail later but the node has 4 inputs
- The first 3 J,K,M are for inputting leaves, the last is for topiary functions

START BY CREATING AN EMPTY PRESET

- Drop down a LSystem Node at the Scene Level
- Drop into the Geometry Level & Select the LSystem
- Let us modify the parameters:
  - **Geometry Tab**
    - Generations 0



LSystem lsystem1

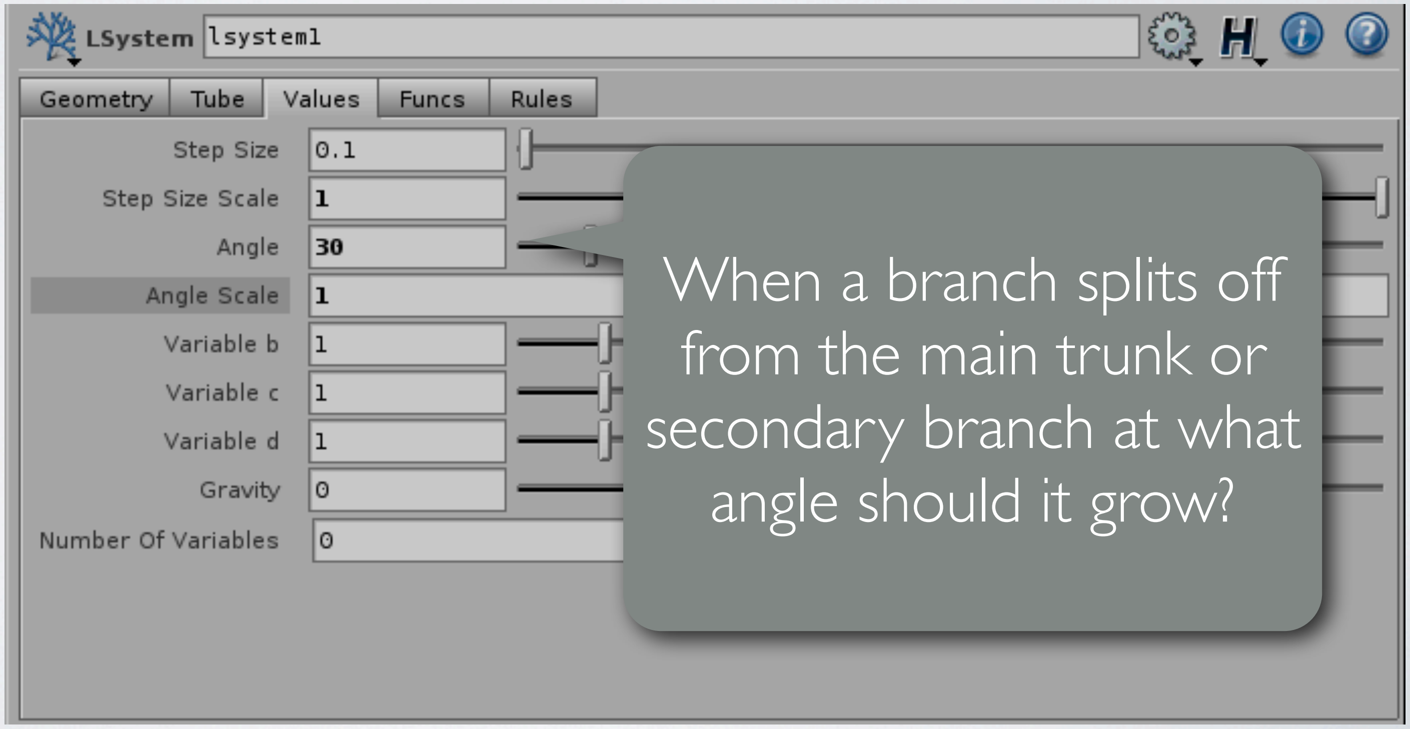
Geometry | Tube | Values | Funcs | Rules

Type: Skeleton
Generations: 0
Start Position: 0
Random Scale: 0
Random Seed: 0
[✓] Continuous Angles
[✓] Continuous Length
[✓] Continuous Width
[ ] Apply Color
Image File:
UV Increment: 0.1
[ ] Point Attributes

Generations or iterations of rules "generated" - Don't understand? Just think of it as how many years old the tree is.

MODIFY THE VALUES TAB

- Let us modify the parameters:
    - **Values** Tab



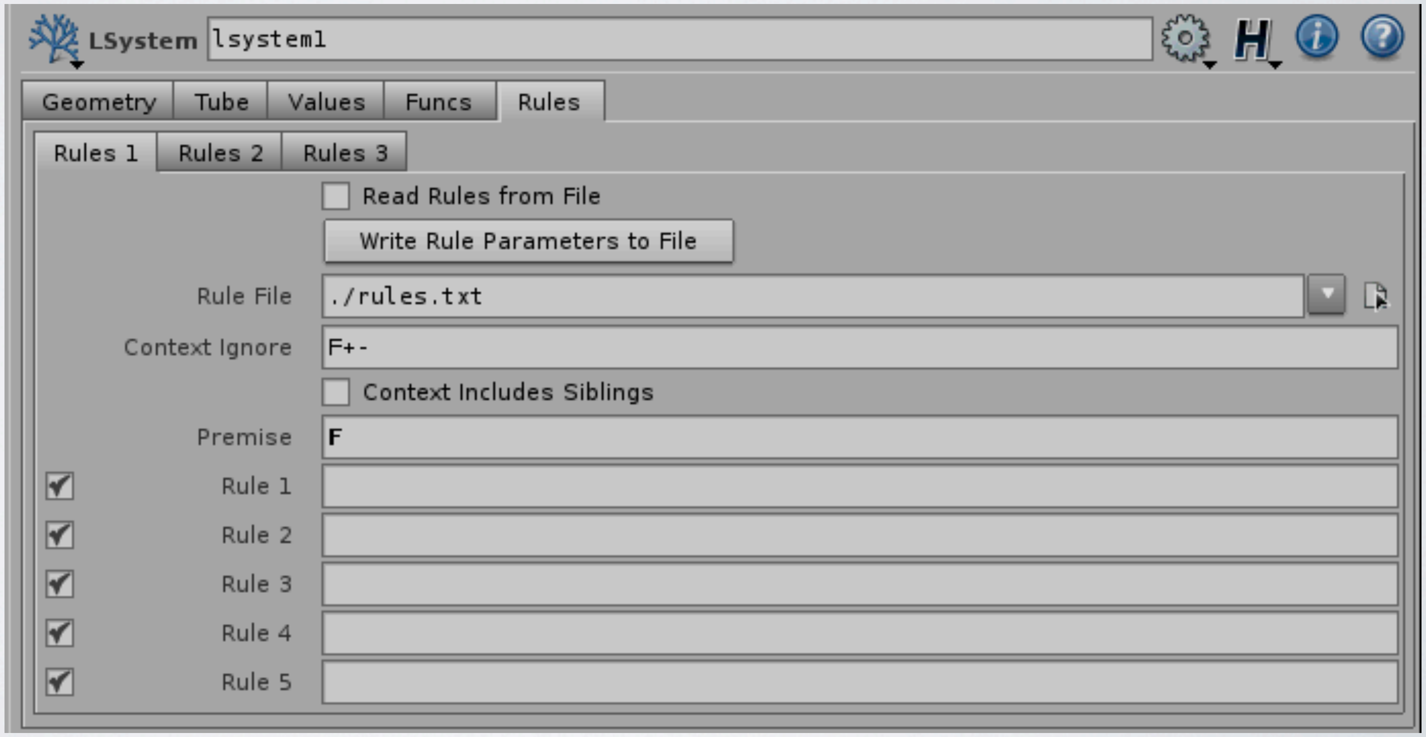
LSystem lsystem1

Step Size 0.1
Step Size Scale 1
Angle 30
Angle Scale 1
Variable b 1
Variable c 1
Variable d 1
Gravity 0
Number Of Variables 0

When a branch splits off from the main trunk or secondary branch at what angle should it grow?

MODIFY THE RULES TAB

- Let us modify the parameters:
  - **Rules** Tab



LSystem lsystem1

Geometry | Tube | Values | Funcs | Rules

Rules 1 | Rules 2 | Rules 3

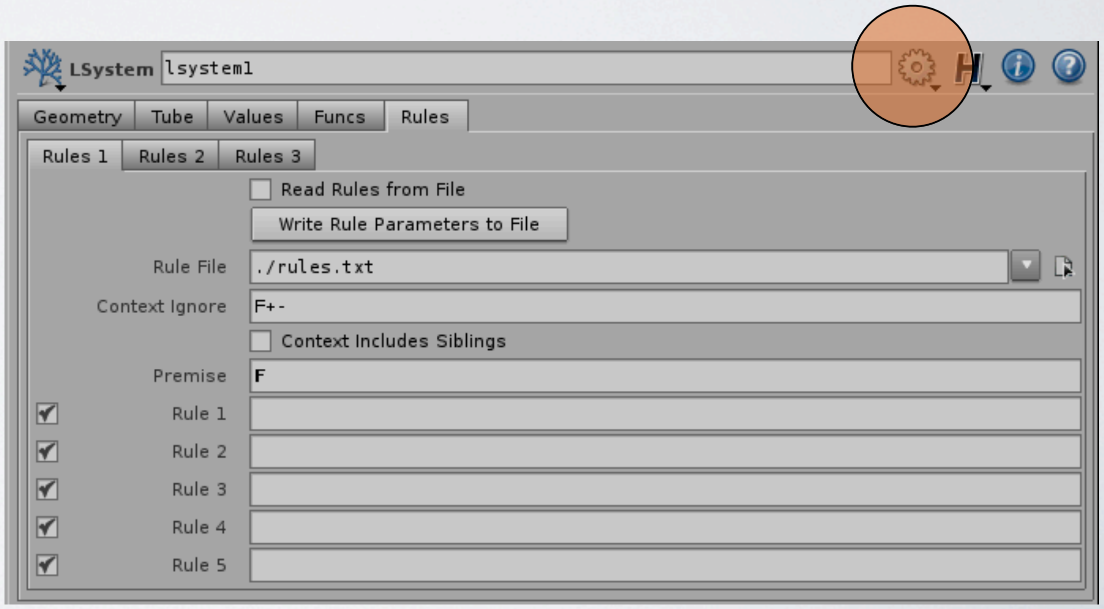
- [ ] Read Rules from File

Write Rule Parameters to File

Rule File	./rules.txt
Context Ignore	F+-
	☐ Context Includes Siblings
Premise	F
✓ Rule 1	
✓ Rule 2	
✓ Rule 3	
✓ Rule 4	
✓ Rule 5	

SAVE AS PRESET - LET'S NAME IT EMPTY LSYSTEM

- Click on the **Gear**
- Select "Save Preset"
- Name it "Empty LSystem"



LSystem lsystem1

☐ Read Rules from File

Rule File ./rules.txt

Context Ignore F+-

☐ Context Includes Siblings

Premise F

✓ Rule 1

✓ Rule 2

✓ Rule 3

✓ Rule 4

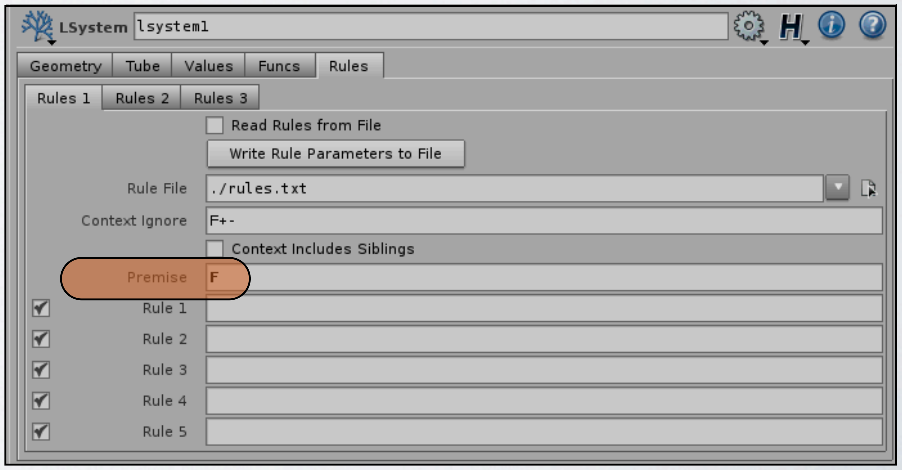
✓ Rule 5

QUICK LOOK AT RULES

- Premise: The "Premise" is the Genetic code of the seed for the plant or tree. It contains the initial information but ***not*** the information for the tree to grow.

In literature the premise is also referred to as the "Initiator"

- Premise: The LSystem at Generation 0.



LSystem lsystem1

Rule File ./rules.txt

Context Ignore F+-

Premise F

Rule 1

Rule 2

Rule 3

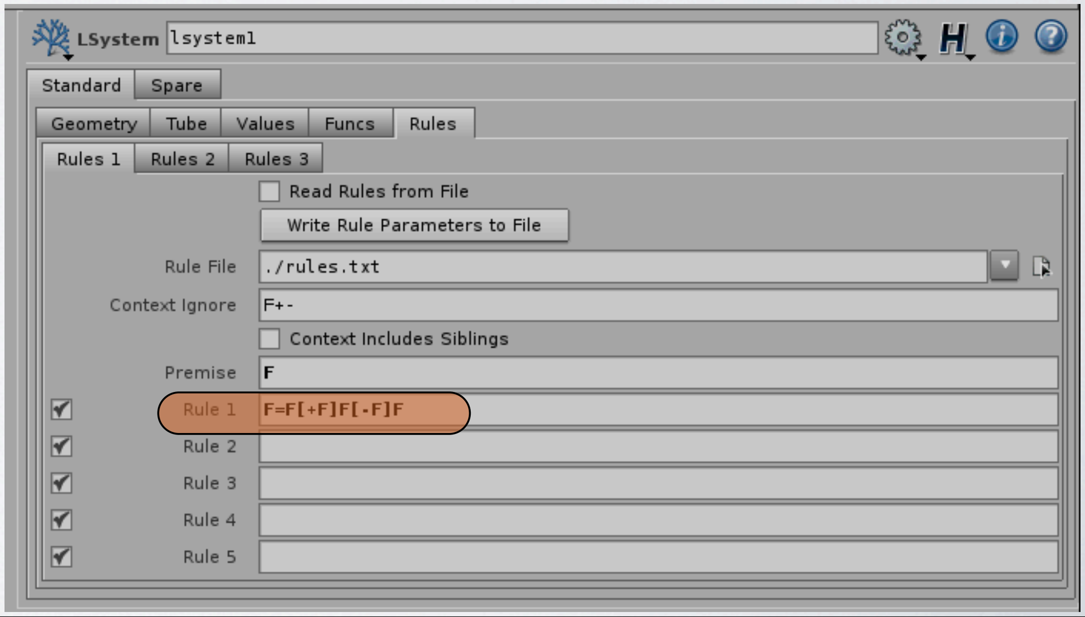
Rule 4

Rule 5

QUICK LOOK AT RULES

In literature the Rules are also referred to as the "Generator"

- Rules: The "Rules" contain the Genetic instructions that are the blue print for the seed (premise) to grow into a plant or tree.



LSystem lsystem1

Standard | Spare
Geometry | Tube | Values | Funcs | Rules
Rules 1 | Rules 2 | Rules 3

☐ Read Rules from File
Write Rule Parameters to File

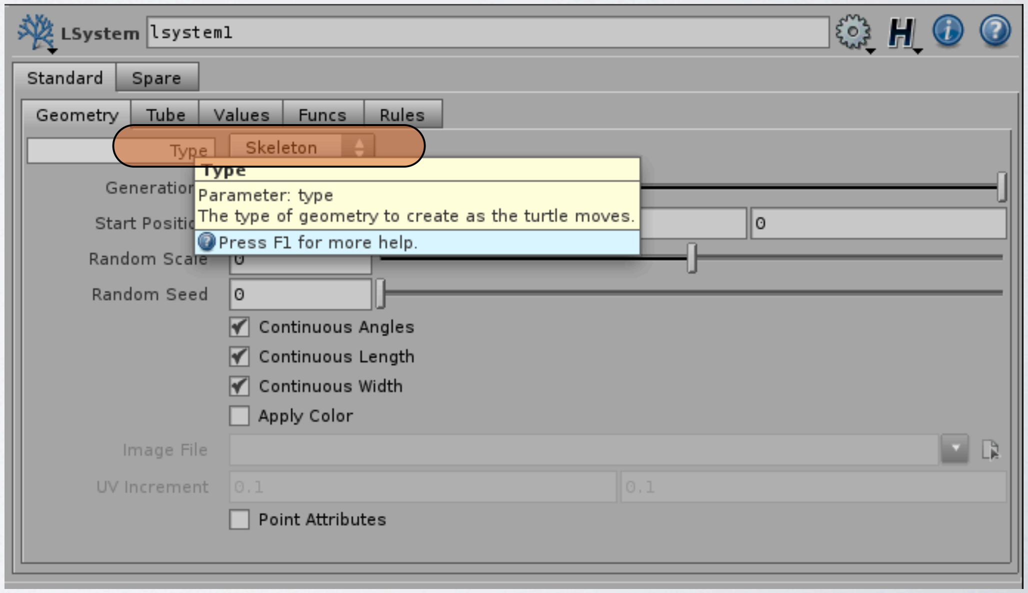
Rule File	./rules.txt
Context Ignore	F+-
	☐ Context Includes Siblings
Premise	F
✓ Rule 1	F=F[+F]F[-F]F
✓ Rule 2	
✓ Rule 3	
✓ Rule 4	
✓ Rule 5	

QUICK LOOK AT GEOMETRY

- Type: LSystems can be rendered as curves (Skeleton) or tubes (Geometry)

Remember Mantra like RSL renders curves

By default branches rendered as curves are rendered at the same thickness. We will solve this problem with an LSystem Command: !(s)



LSystem lsystem1

Standard | Spare

Geometry | Tube | Values | Funcs | Rules

Type: Skeleton

Type
Parameter: type
The type of geometry to create as the turtle moves.
Press F1 for more help.

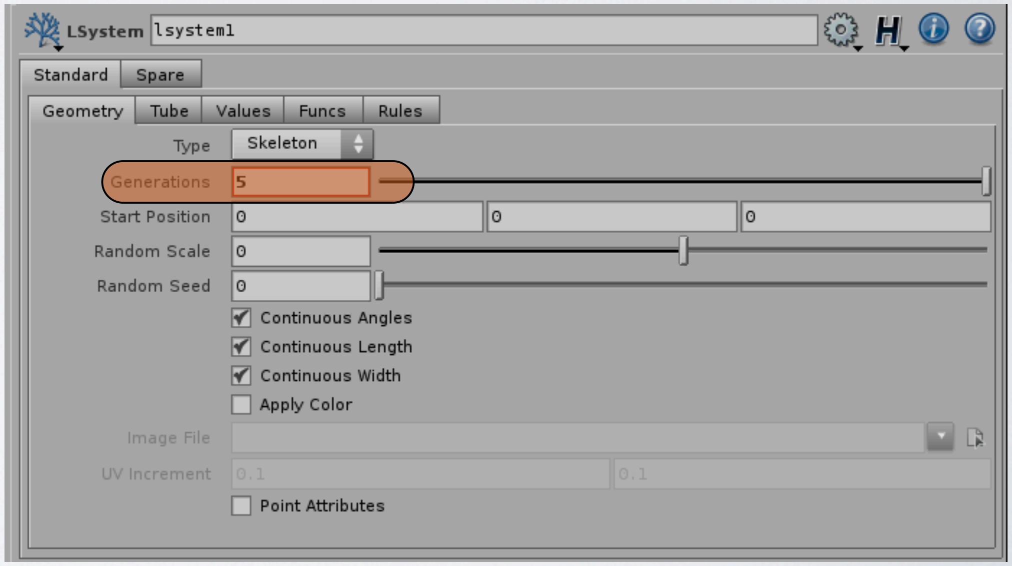
Generation
Start Position ... 0
Random Scale 0
Random Seed 0
☑ Continuous Angles
☑ Continuous Length
☑ Continuous Width
☐ Apply Color
Image File
UV Increment 0.1 0.1
☐ Point Attributes

QUICK LOOK AT GEOMETRY

LSystems is calculation heavy. Keep generations low.

- Generations: Increasing the value in Generations grows the tree from sapling to mature tree.

When debugging. Keep your generations to a whole number.



LSystem lsystem1

Standard | Spare

Geometry | Tube | Values | Funcs | Rules

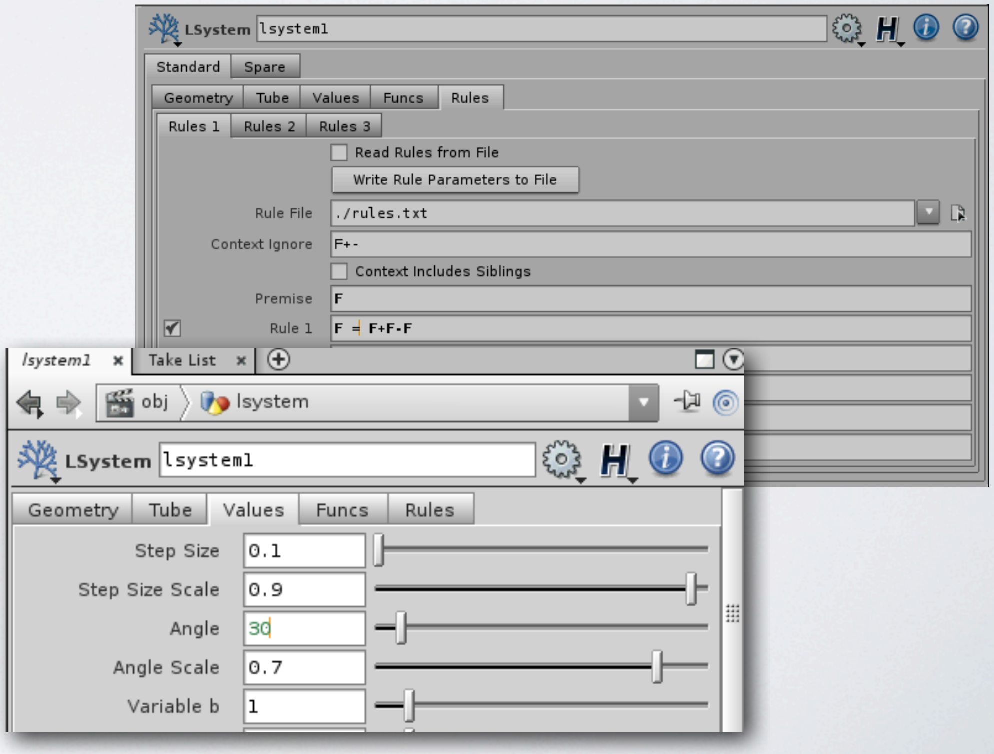
Type: Skeleton
Generations: 5
Start Position: 0 0 0
Random Scale: 0
Random Seed: 0
✓ Continuous Angles
✓ Continuous Length
✓ Continuous Width
☐ Apply Color
Image File:
UV Increment: 0.1 0.1
☐ Point Attributes

FIRST COMMANDS

2D Systems, also known as Planar Turning

- F - Turtle forward one step while drawing ink
- + turn right "x" degrees
- - turn left "x" degrees

How do you define degrees? It's in the "Values" Tab.



LSystem lsystem1

☐ Read Rules from File

Rule File ./rules.txt

Context Ignore F+-

☐ Context Includes Siblings

Premise F

✓ Rule 1 F = F+F-F

LSystem lsystem1

Step Size 0.1

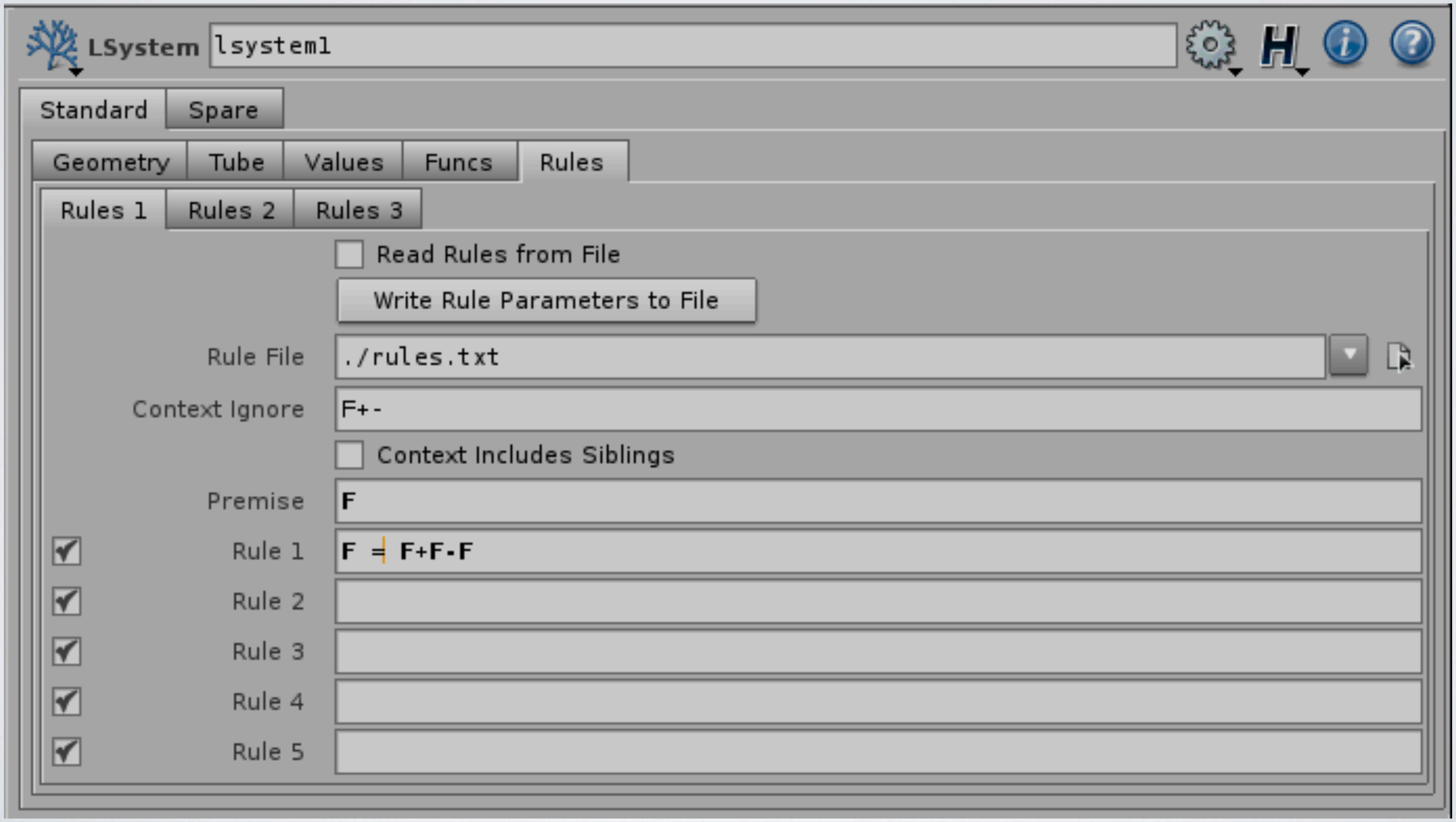
Step Size Scale 0.9

Angle 30

Angle Scale 0.7

Variable b 1

FIRST LSYSTEM



LSystem lsystem1

Standard | Spare

Geometry | Tube | Values | Funcs | Rules

Rules 1 | Rules 2 | Rules 3

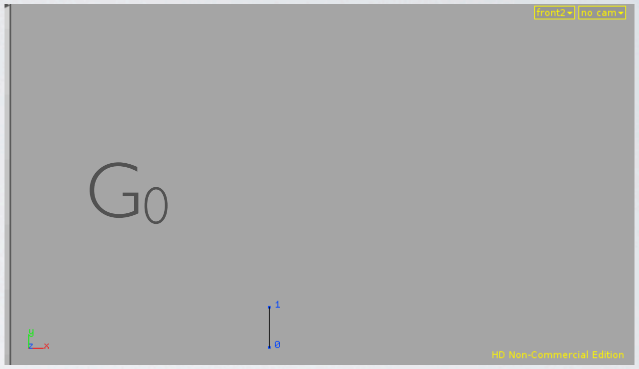
☐ Read Rules from File

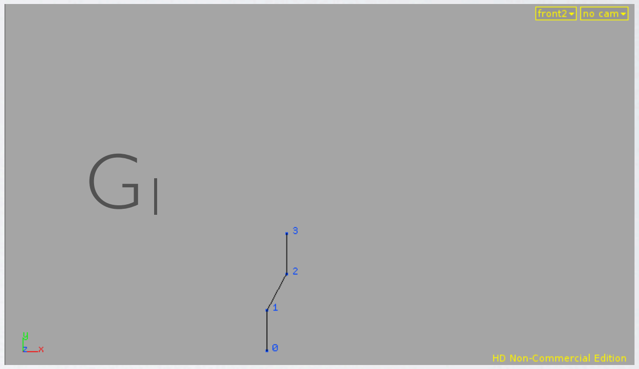
Write Rule Parameters to File

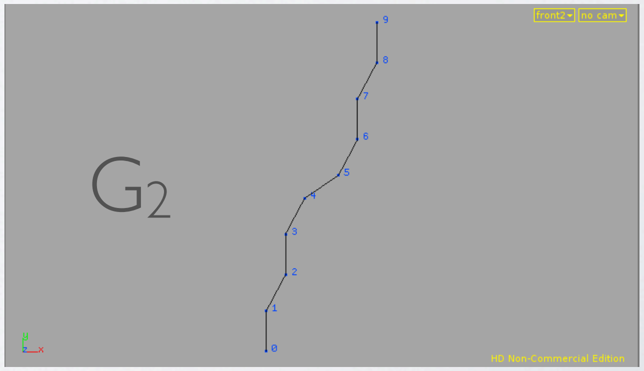
Rule File	./rules.txt
Context Ignore	F+-
	☐ Context Includes Siblings
Premise	F
✓ Rule 1	F = F+F-F
✓ Rule 2	
✓ Rule 3	
✓ Rule 4	
✓ Rule 5	

WHAT HAPPENED?

- Premise: F
- Rule 1: F = F+F-F
- G_0: F
- G_1: F+F-F
- G_2: F+F-F+F+F-F-F+F-F





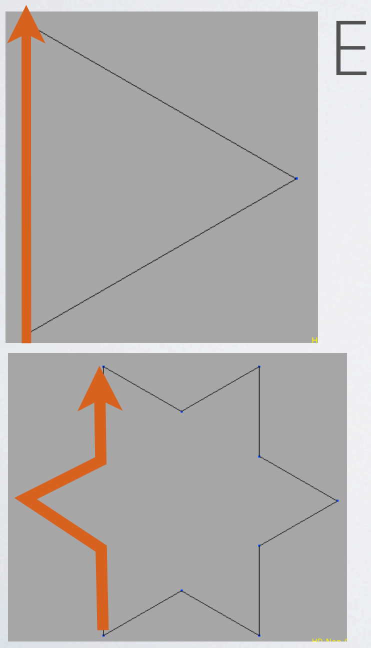


BESIDES F

- F - Move forward one step and draw
- f - Move forward one step and do not draw
- H - Move forward half a step and draw
- h - Move forward half a step and do not draw

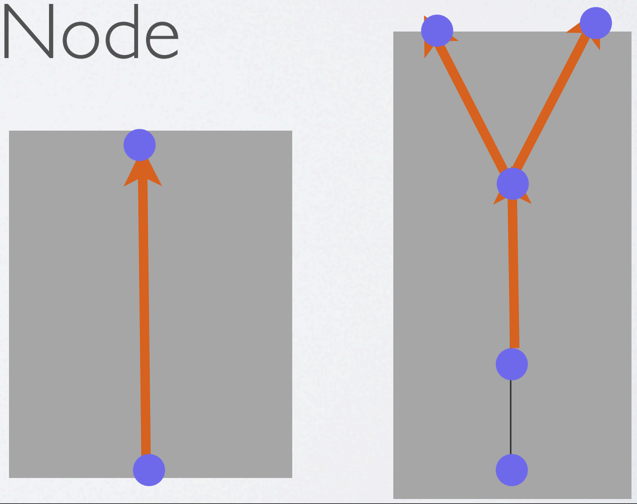
EDGE VS NODE REWRITING

In the case of edge rewriting, productions substitute figures for polygon edges, while in node rewriting, productions operate on polygon vertices. Both approaches rely on capturing the recursive structure of figures and relating it to a tiling of a plane.



Edge

Node

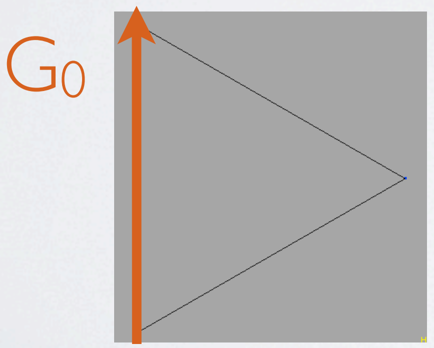


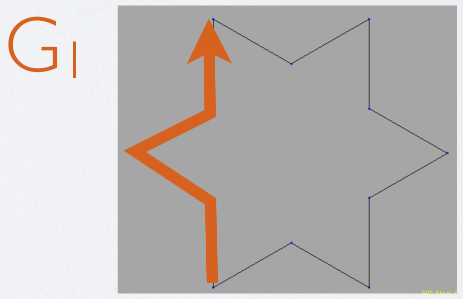
EDGE REWRITING

Also known as Edge Replacement

Productions substitute figures for polygon edges

Example: Koch Snow Flake





KOCH SNOW FLAKE

Good Example of Edge Replacement

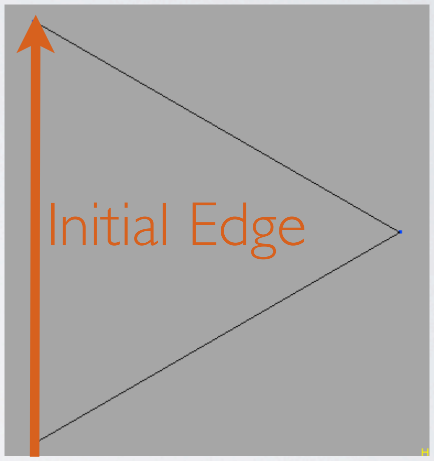
Remember the sum of the angles of a triangle add up to 360 degrees

Angle: 60

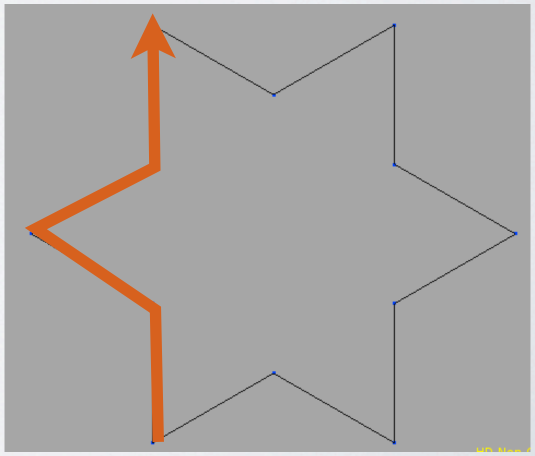
Premise: F++F++F

Rule 1: F = F-F++F-F

G_0:



G_1:

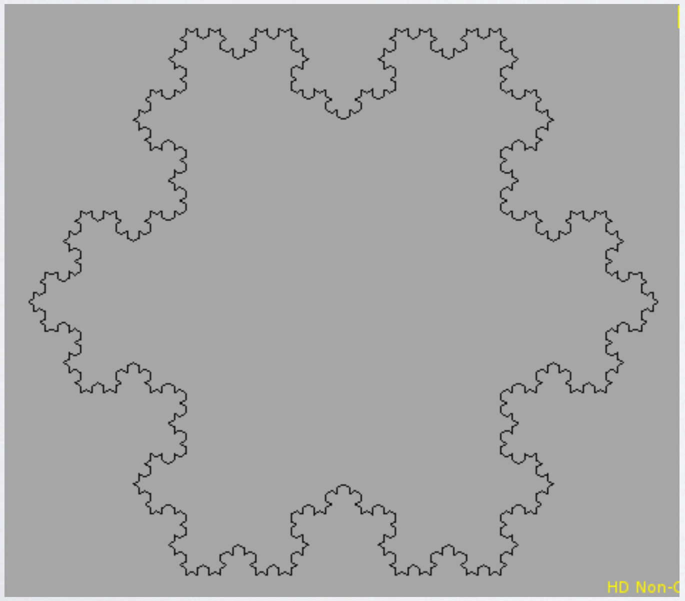


KOCH SNOW FLAKE

Notice the Snow Flake gets Larger with Each Generation

Why is that?

G_4:



WHAT HAPPENS WHEN?

You reverse the +,- commands in Rule 1

Give it a try!

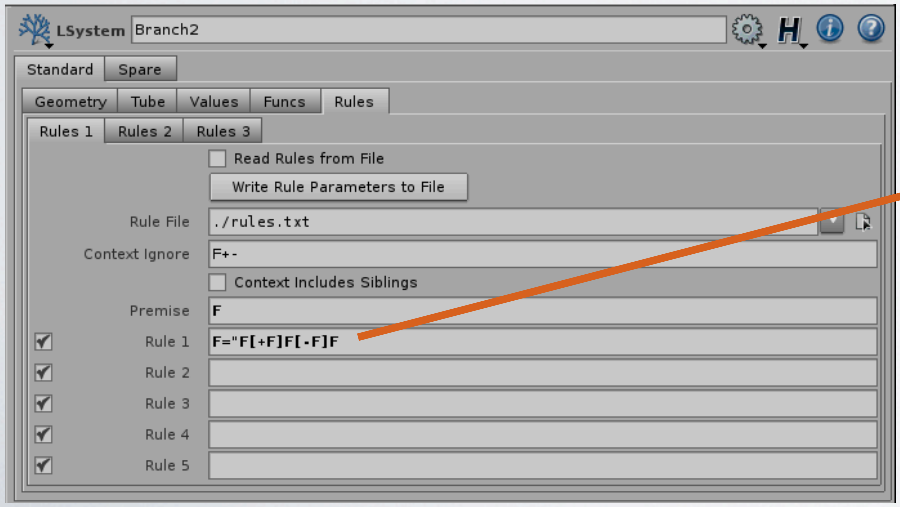
Angle: 60

Premise: F++F++F

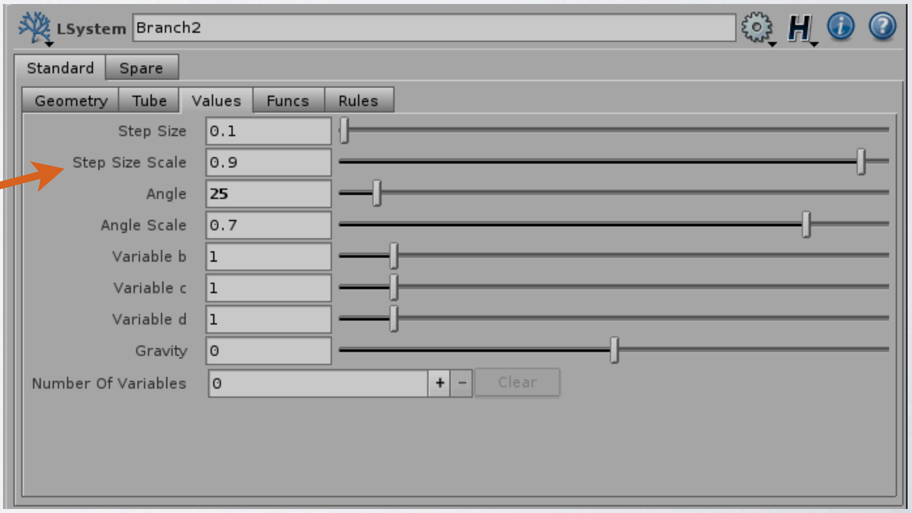
Rule 1: F = F+F--F+F

NEXT COMMANDS - BRANCHING AND STEP

- **[ ]** - Anything within **Square Brackets** is a unique branch
- **"** - **scales** growth between generations. As a tree grows the older branches are usually longer then the newer branches. The value is tied to the "**Step Size Scale**" in the **Values Tab**



```
Branch2
[ ] Read Rules from File
Rule File        ./rules.txt
Context Ignore   F+-
[ ] Context Includes Siblings
Premise          F
[✓] Rule 1       F="F[+F]F[-F]F
[✓] Rule 2
[✓] Rule 3
[✓] Rule 4
[✓] Rule 5
```

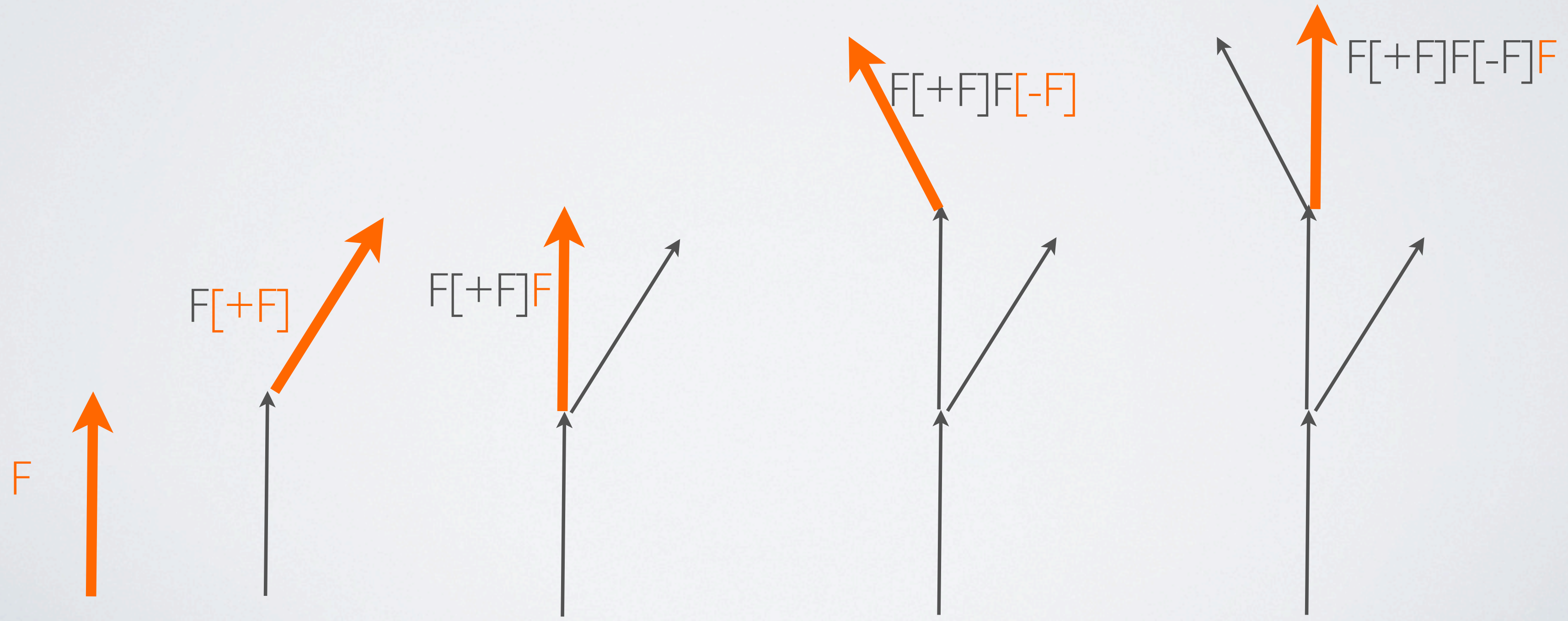


```
Branch2
Step Size            0.1
Step Size Scale      0.9
Angle                25
Angle Scale          0.7
Variable b           1
Variable c           1
Variable d           1
Gravity              0
Number Of Variables  0
```

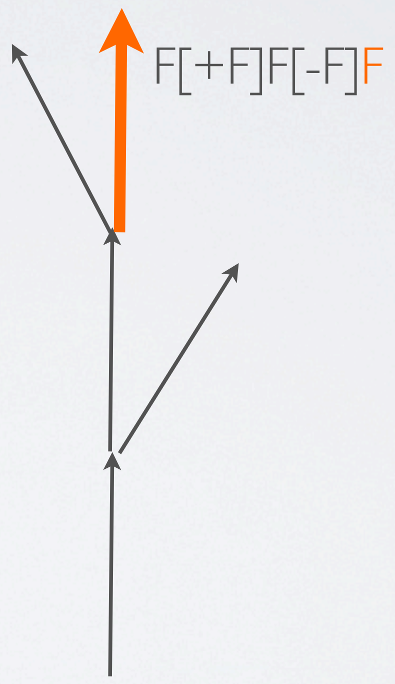
BRANCHING EXAMPLE

- **Premise:** F
- **Rule 1:** F = F[+F]F[-F]F

For programmers - You can think of [ ] as the push and pop states of graphics programming



BRANCHES "POP" BACK



Branches after the "] " command always go back to their starting point.

BRANCHING EXAMPLE

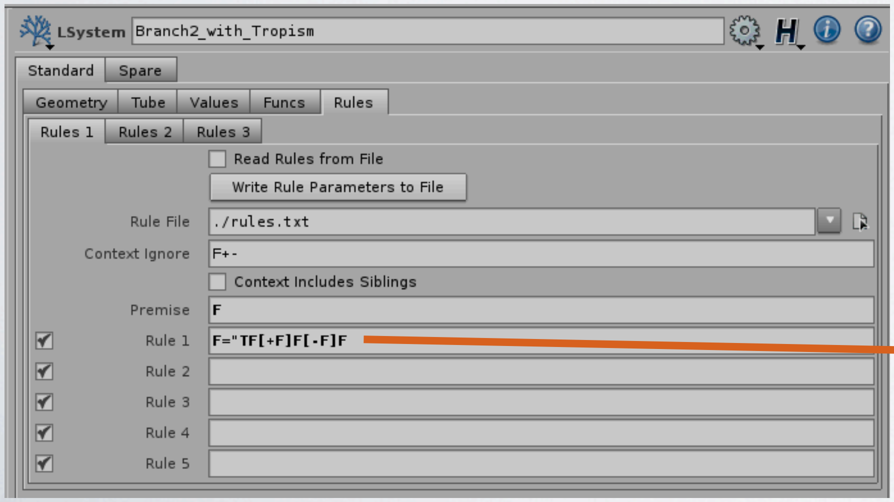
With Scaling

- **Premise:** F
- **Rule 1:** F = “F[+F]F[-F]F **or**
- **Rule 1:** F = “(0.8)F[+F]F[-F]F

You can either use the “Step Size Scale” in the Values tab or directly set the value in the rules using (x) right after the “ command

TROPISM (GRAVITY BENDS LIMBS)

- **T** -T is the command to bend branches due to controlled by the "**Gravity**" parameter in the V
- **Notice -** The Main trunk is **not** affected by tr branch is vertically straight tropism has no effect



LSystem Branch2_with_Tropism

Standard | Spare

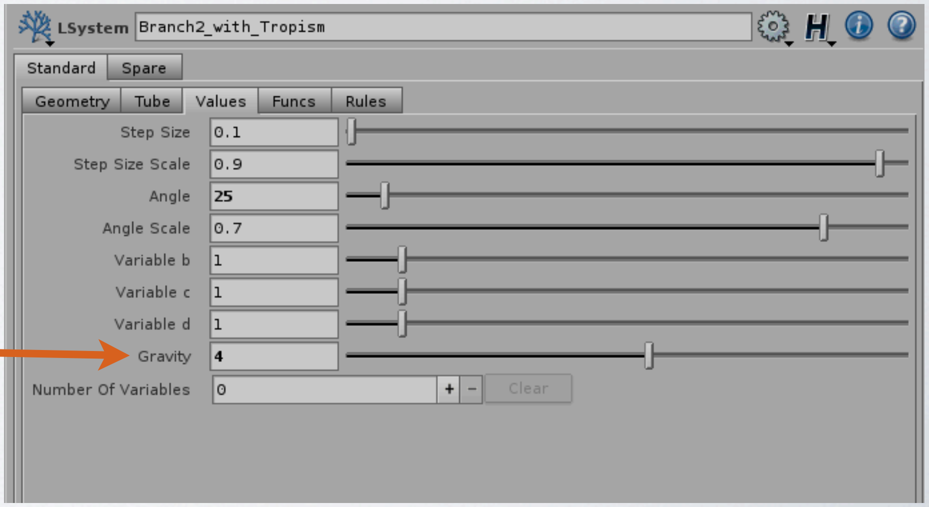
Geometry | Tube | Values | Funcs | Rules

Rules 1 | Rules 2 | Rules 3

☐ Read Rules from File

Write Rule Parameters to File

Rule File	./rules.txt
Context Ignore	F+-
	☐ Context Includes Siblings
Premise	F
✓ Rule 1	F="TF[+F]F[-F]F
✓ Rule 2	
✓ Rule 3	
✓ Rule 4	
✓ Rule 5	



LSystem Branch2_with_Tropism

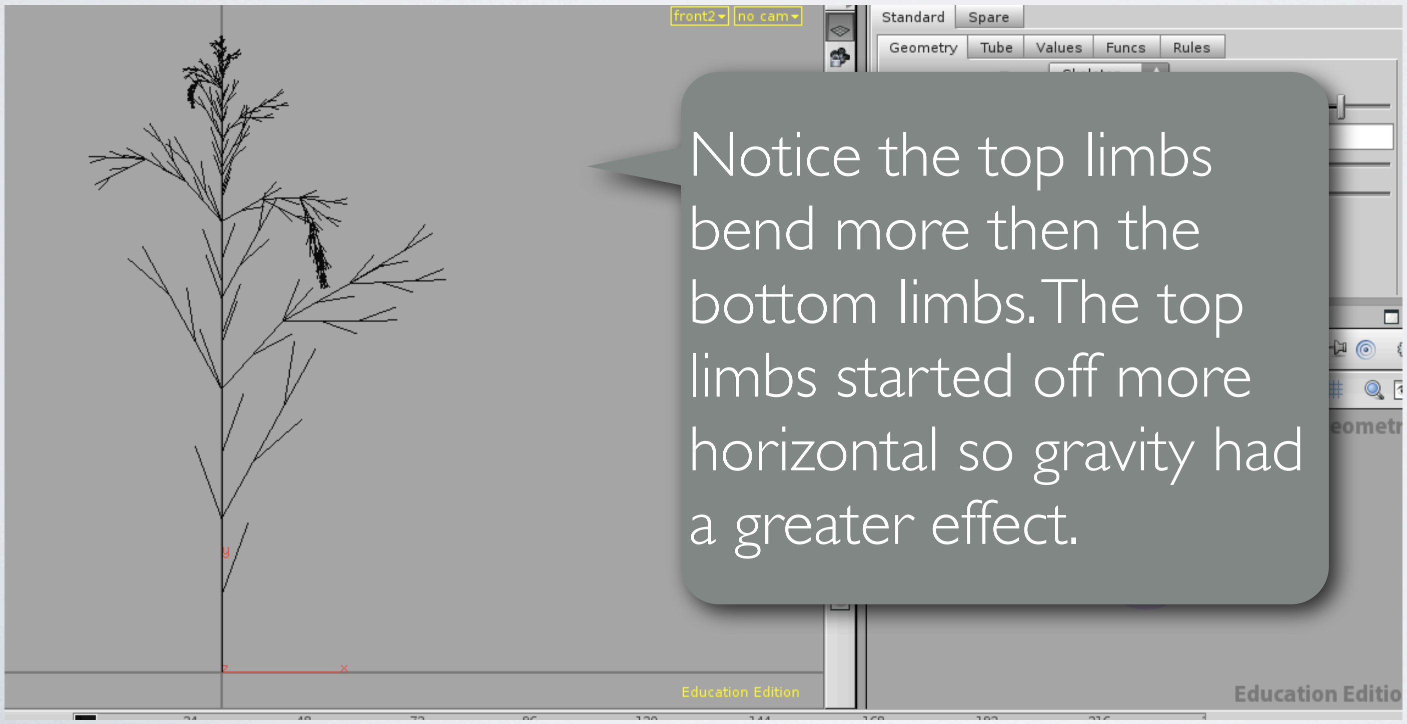
Standard | Spare

Geometry | Tube | Values | Funcs | Rules

Step Size	0.1
Step Size Scale	0.9
Angle	25
Angle Scale	0.7
Variable b	1
Variable c	1
Variable d	1
Gravity	4
Number Of Variables	0

TROPISM EXAMPLE

- **Premise:** F
- **Rule 1** - “TF[+F]F[-F]F

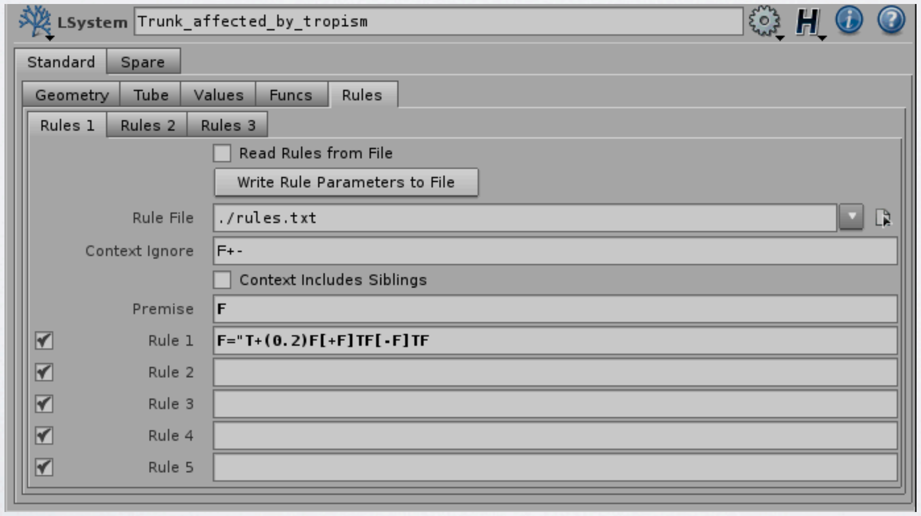


Notice the top limbs bend more then the bottom limbs. The top limbs started off more horizontal so gravity had a greater effect.

SO HOW DO I BEND THE MAIN TRUNK?

- **( )** - You can hard code a value to an operator by appending (n). Therefore by adding a small turn angle to the main trunk it will be affected by tropism.

- **Notice -** The Main trunk is **affected** by tropism. Notice the top of the trunk is more affected then the bottom.



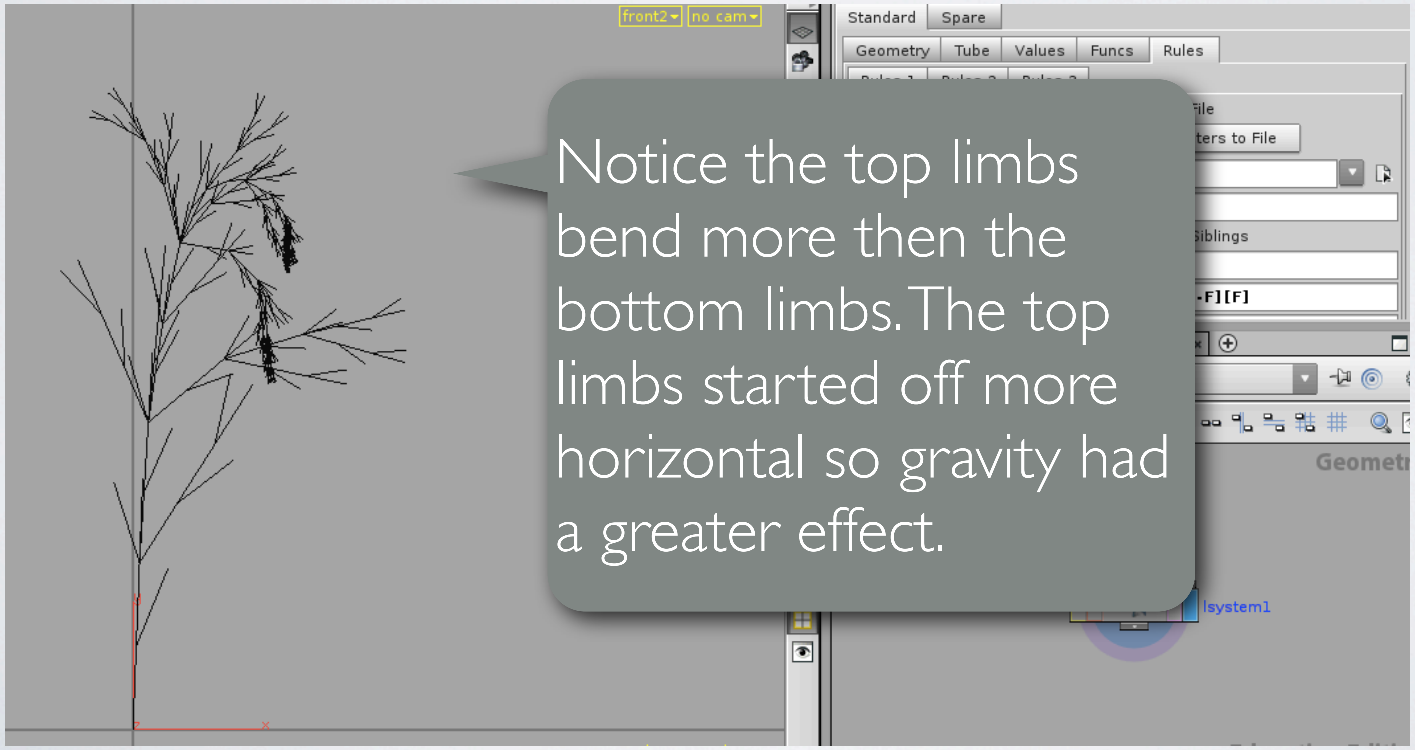
LSystem Trunk_affected_by_tropism

☐ Read Rules from File

Rule File	./rules.txt
Context Ignore	F+-
	☐ Context Includes Siblings
Premise	F
✓ Rule 1	F="T+(0.2)F[+F]TF[-F]TF
✓ Rule 2	
✓ Rule 3	
✓ Rule 4	
✓ Rule 5	

TROPISM EXAMPLE - BENDING TRUNK

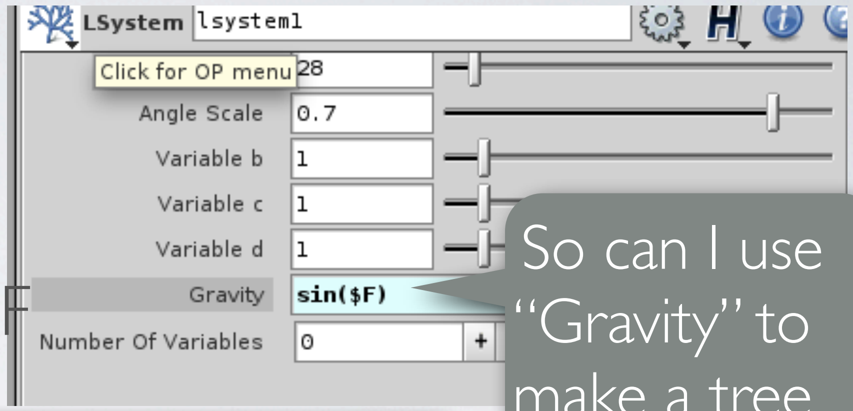
- **Premise:** F
- **Rule 1** - "T+(0.3)F[+F]F[-F]F



Notice the top limbs bend more then the bottom limbs. The top limbs started off more horizontal so gravity had a greater effect.

MAKING A TREE SWAY

- **Premise:** F
- **Rule 1** - "T+(0.3)F[+F]F[-F]F



LSystem lsystem1

Click for OP menu 28

Angle Scale 0.7

Variable b 1

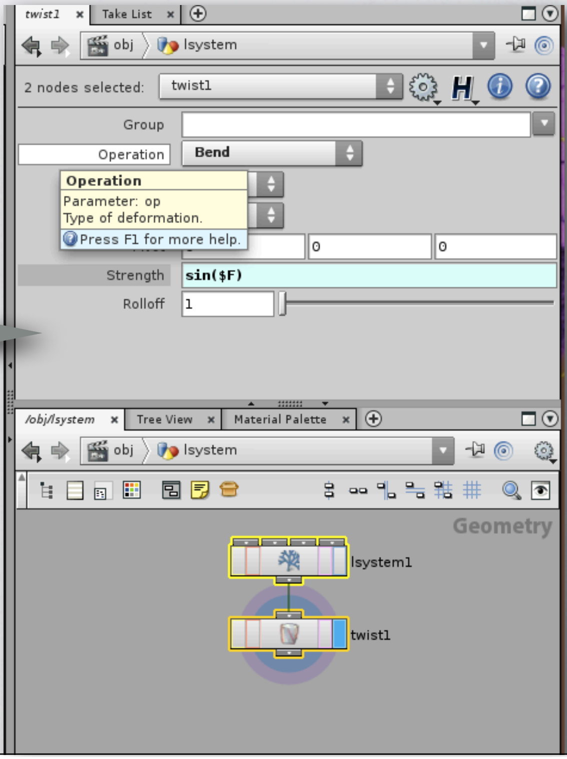
Variable c 1

Variable d 1

Gravity sin($F)

Number Of Variables 0

So can I use "Gravity" to make a tree sway?



2 nodes selected: twist1

Group

Operation Bend

Operation
Parameter: op
Type of deformation.
Press F1 for more help.

0 0

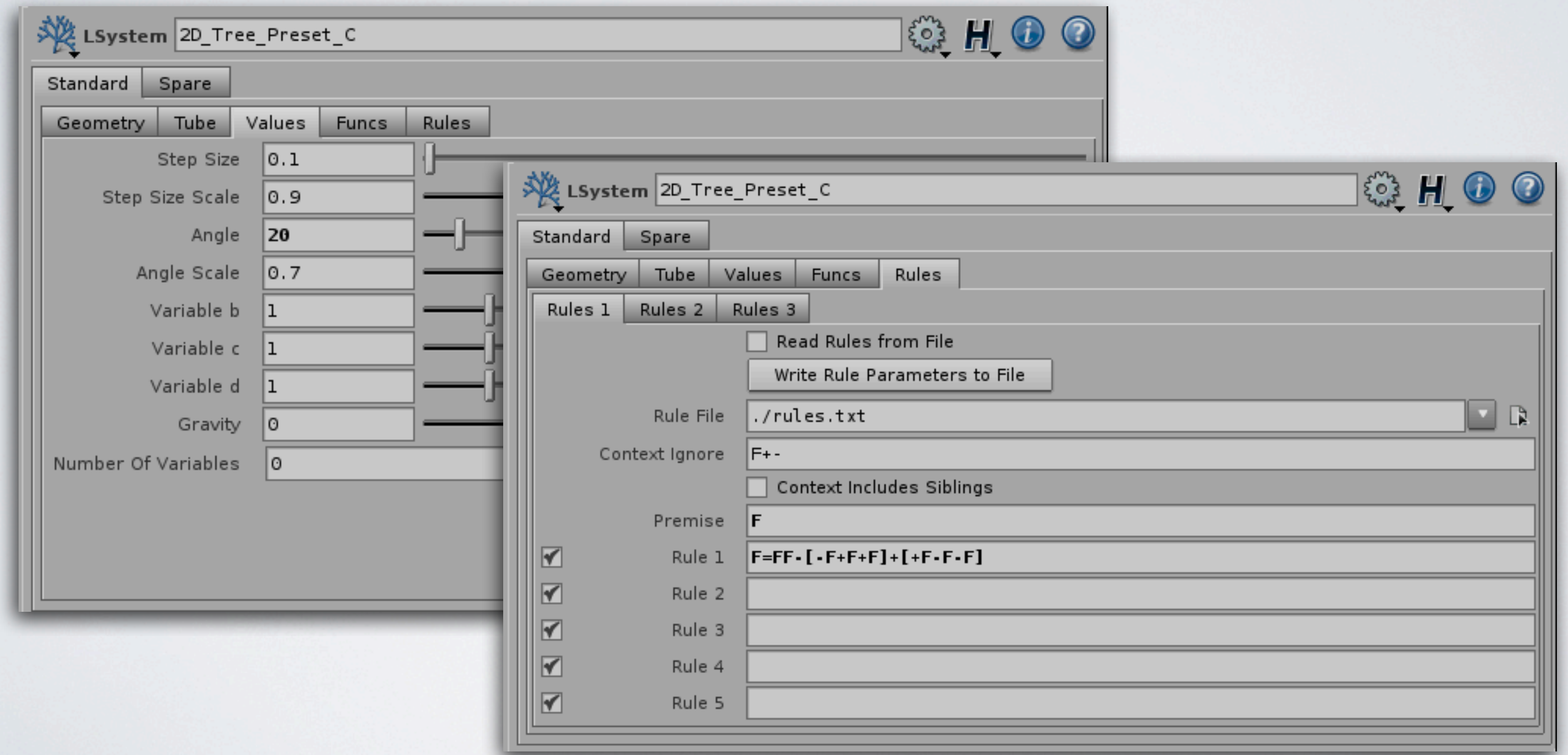
Strength sin($F)

Rolloff 1

Not Really? Use a **Twist** Sop instead.

CHALLENGE

Draw the tree on paper before using Houdini



LSystem 2D_Tree_Preset_C

Standard | Spare

Geometry | Tube | Values | Funcs | Rules

Step Size	0.1
Step Size Scale	0.9
Angle	20
Angle Scale	0.7
Variable b	1
Variable c	1
Variable d	1
Gravity	0
Number Of Variables	0

LSystem 2D_Tree_Preset_C

Standard | Spare

Geometry | Tube | Values | Funcs | Rules

Rules 1 | Rules 2 | Rules 3

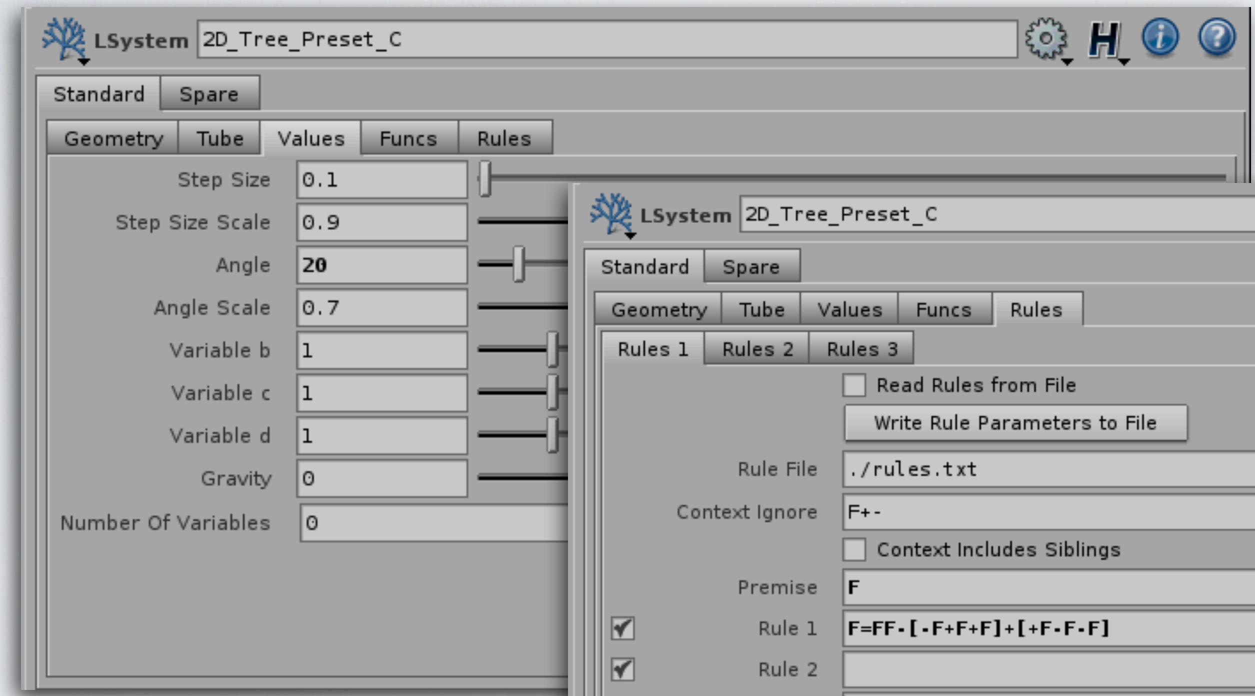
☐ Read Rules from File

Write Rule Parameters to File

	Rule File	./rules.txt
	Context Ignore	F+-
	☐ Context Includes Siblings	
	Premise	F
✓	Rule 1	F=FF-[-F+F+F]+[+F-F-F]
✓	Rule 2	
✓	Rule 3	
✓	Rule 4	
✓	Rule 5	

CHALLENGE

Answer is Preset - "2D Plant C"

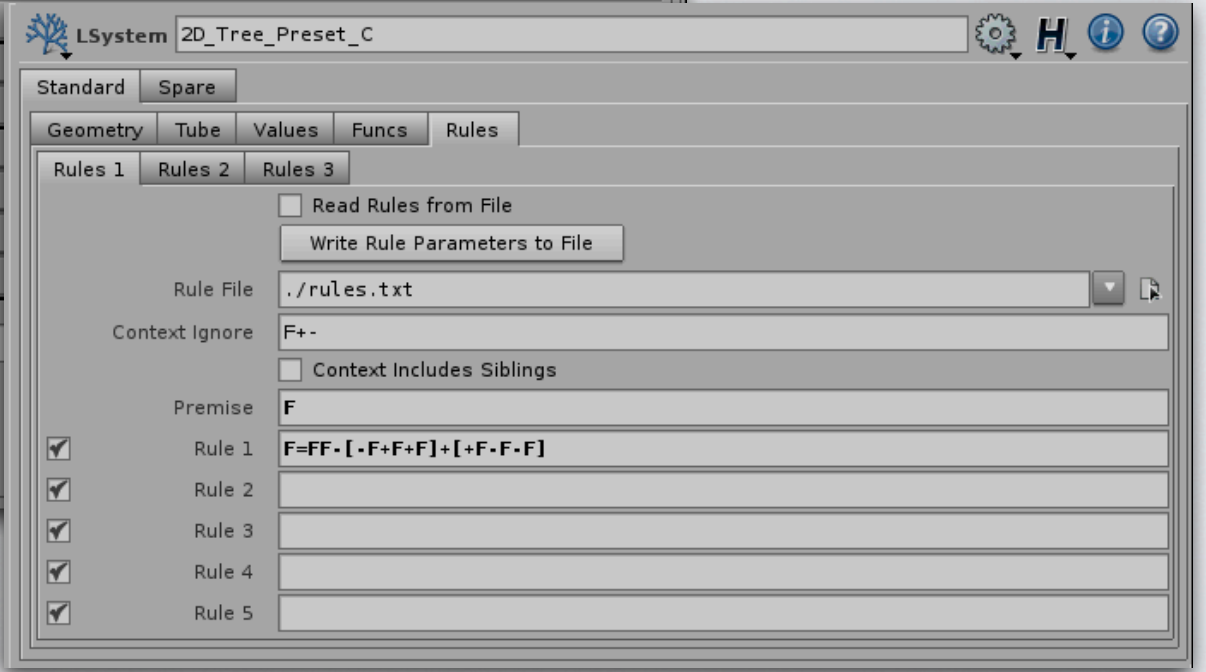


LSystem 2D_Tree_Preset_C

Standard | Spare

Geometry | Tube | Values | Funcs | Rules

Step Size	0.1
Step Size Scale	0.9
Angle	20
Angle Scale	0.7
Variable b	1
Variable c	1
Variable d	1
Gravity	0
Number Of Variables	0



LSystem 2D_Tree_Preset_C

Standard | Spare

Geometry | Tube | Values | Funcs | Rules

Rules 1 | Rules 2 | Rules 3

☐ Read Rules from File

Write Rule Parameters to File

Rule File	./rules.txt
Context Ignore	F+-
☐ Context Includes Siblings	
Premise	F
✓ Rule 1	F=FF-[-F+F+F]+[+F-F-F]
✓ Rule 2	
✓ Rule 3	
✓ Rule 4	
✓ Rule 5	

NODE REWRITING

- Up to now we have done the simplest form of LSystems. **Edge Replacement**. Now we will look at the next level of complexity **Node Replacement.**

NODE REWRITING

Also known as Appending

The idea of node rewriting is to substitute new polygons for nodes (points) of the predecessor curve.

As an example:

Premise: FX
Rule 1: X = F[+FX][-FX]

Notice: Here we use "X" not as a command like "F" but as a variable. "X" does not draw

NODE REWRITING

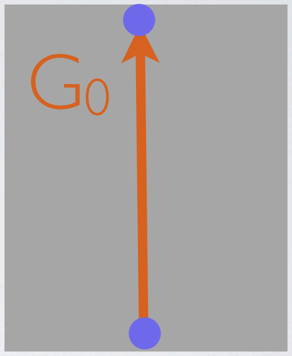
Premise: FX

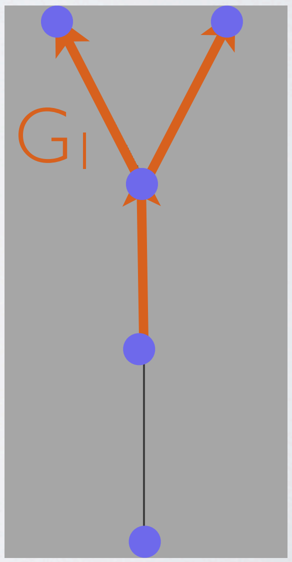
Rule 1: X = F[+FX][-FX]

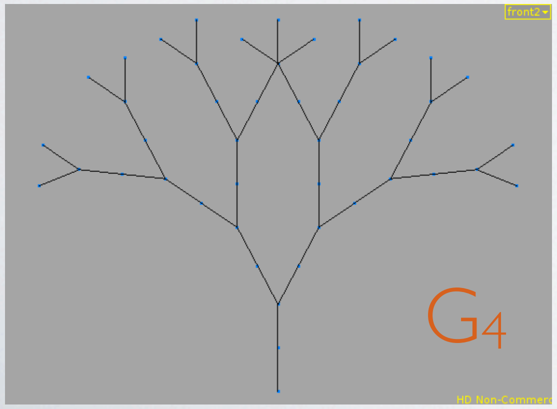
G_0 : F

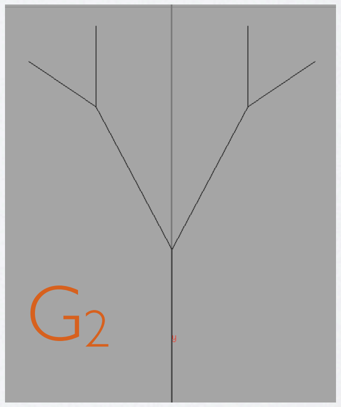
G_1 : F[+FX][-FX]

G_2 : F[+FXF[+FX][-FX]][-FF[+FX][-FX]]

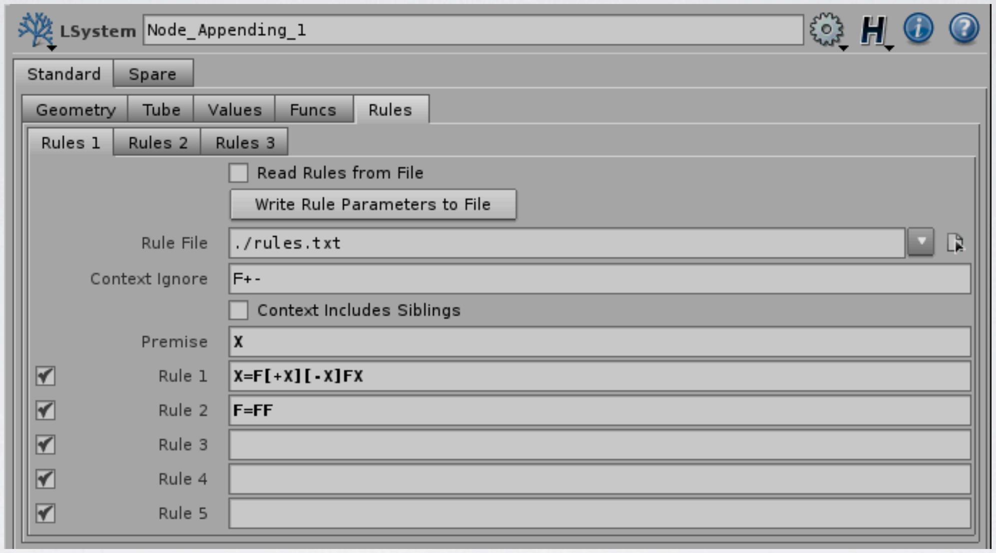








MULTIPLE RULES



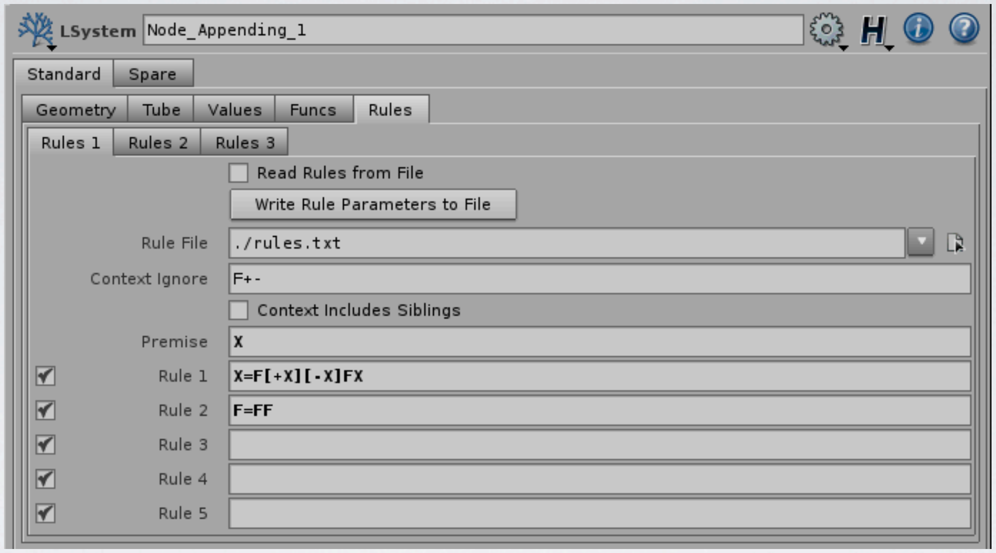
Node_Appending_1

☐ Read Rules from File

Rule File	./rules.txt
Context Ignore	F+-
	☐ Context Includes Siblings
Premise	X
✓ Rule 1	X=F[+X][-X]FX
✓ Rule 2	F=FF
✓ Rule 3	
✓ Rule 4	
✓ Rule 5	

- Do not replace a variable twice in one generation
- When confused use **opinfo** command in **HScript Textport**
- opinfo -v /obj/geo1/lsystem1/

MULTIPLE RULES



LSystem: Node_Appending_1

[ ] Read Rules from File

Rule File: ./rules.txt

Context Ignore: F+-

[ ] Context Includes Siblings

Premise: X

[✓] Rule 1: X=F[+X][-X]FX

[✓] Rule 2: F=FF

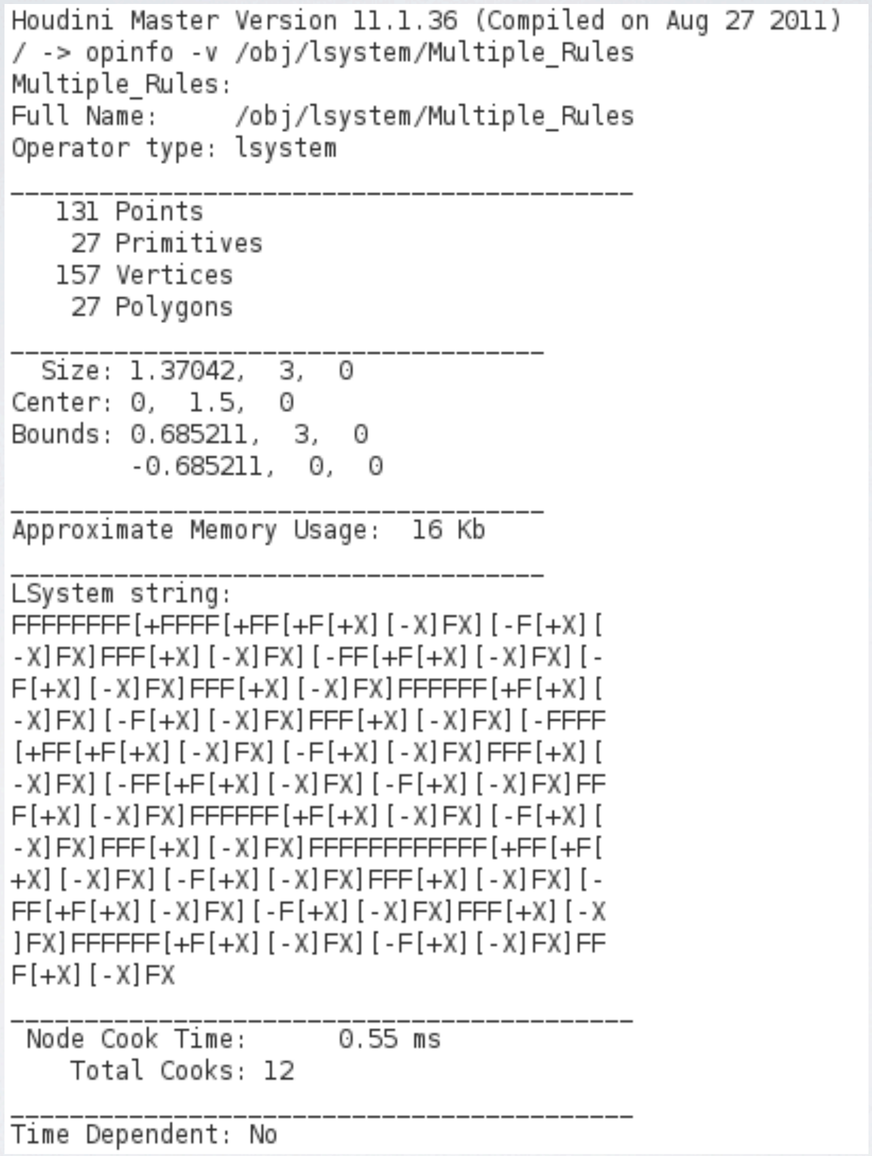
[✓] Rule 3:

[✓] Rule 4:

[✓] Rule 5:

- G0: X (Nothing is drawn)
- G1: F[+X][-X]FX
- G2: FF[+ F[+X][-X]FX][-F[+X][-X]FX]FFF[+X][-X]FX

MULTIPLE RULES



```
Houdini Master Version 11.1.36 (Compiled on Aug 27 2011)
/ -> opinfo -v /obj/lsystem/Multiple_Rules
Multiple_Rules:
Full Name:     /obj/lsystem/Multiple_Rules
Operator type: lsystem

__________________________________________
   131 Points
    27 Primitives
   157 Vertices
    27 Polygons

____________________________________
  Size: 1.37042,  3,  0
Center: 0,  1.5,  0
Bounds: 0.685211,  3,  0
        -0.685211,  0,  0

____________________________________
Approximate Memory Usage:  16 Kb

____________________________________
LSystem string:
FFFFFFFF[+FFFF[+FF[+F[+X][-X]FX][-F[+X][
-X]FX]FFF[+X][-X]FX][-FF[+F[+X][-X]FX][-
F[+X][-X]FX]FFF[+X][-X]FX]FFFFFF[+F[+X][
-X]FX][-F[+X][-X]FX]FFF[+X][-X]FX][-FFFF
[+FF[+F[+X][-X]FX][-F[+X][-X]FX]FFF[+X][
-X]FX][-FF[+F[+X][-X]FX][-F[+X][-X]FX]FF
F[+X][-X]FX]FFFFFF[+F[+X][-X]FX][-F[+X][
-X]FX]FFF[+X][-X]FX]FFFFFFFFFFFF[+FF[+F[
+X][-X]FX][-F[+X][-X]FX]FFF[+X][-X]FX][-
FF[+F[+X][-X]FX][-F[+X][-X]FX]FFF[+X][-X
]FX]FFFFFF[+F[+X][-X]FX][-F[+X][-X]FX]FF
F[+X][-X]FX

__________________________________________
 Node Cook Time:      0.55 ms
    Total Cooks: 12

__________________________________________
Time Dependent: No
```

CONTROLLING THICKNESS, LENGTH, AND ANGLE OF BRANCHES

- ! - multiply current thickness by s. (s) - Default thickness scale
- “ - multiply current length by s. (s) - Default step size scale
- ; - multiply current angle by s. Default angle scale.

Thickness, and **Thickness Scale** can be found in the **Tube** tab

ADDING RANDOMNESS TO BRANCH ANGLE

- ~(a) - Pitch/Roll/Turn random amount up to “a” degrees.

3D

At last we start to build real trees

- +,- Rotate Right, Rotate Left
- &,^ Pitch Up, Pitch Down
- \,/ Roll Clockwise, Counter Clockwise

PROBABILITIES

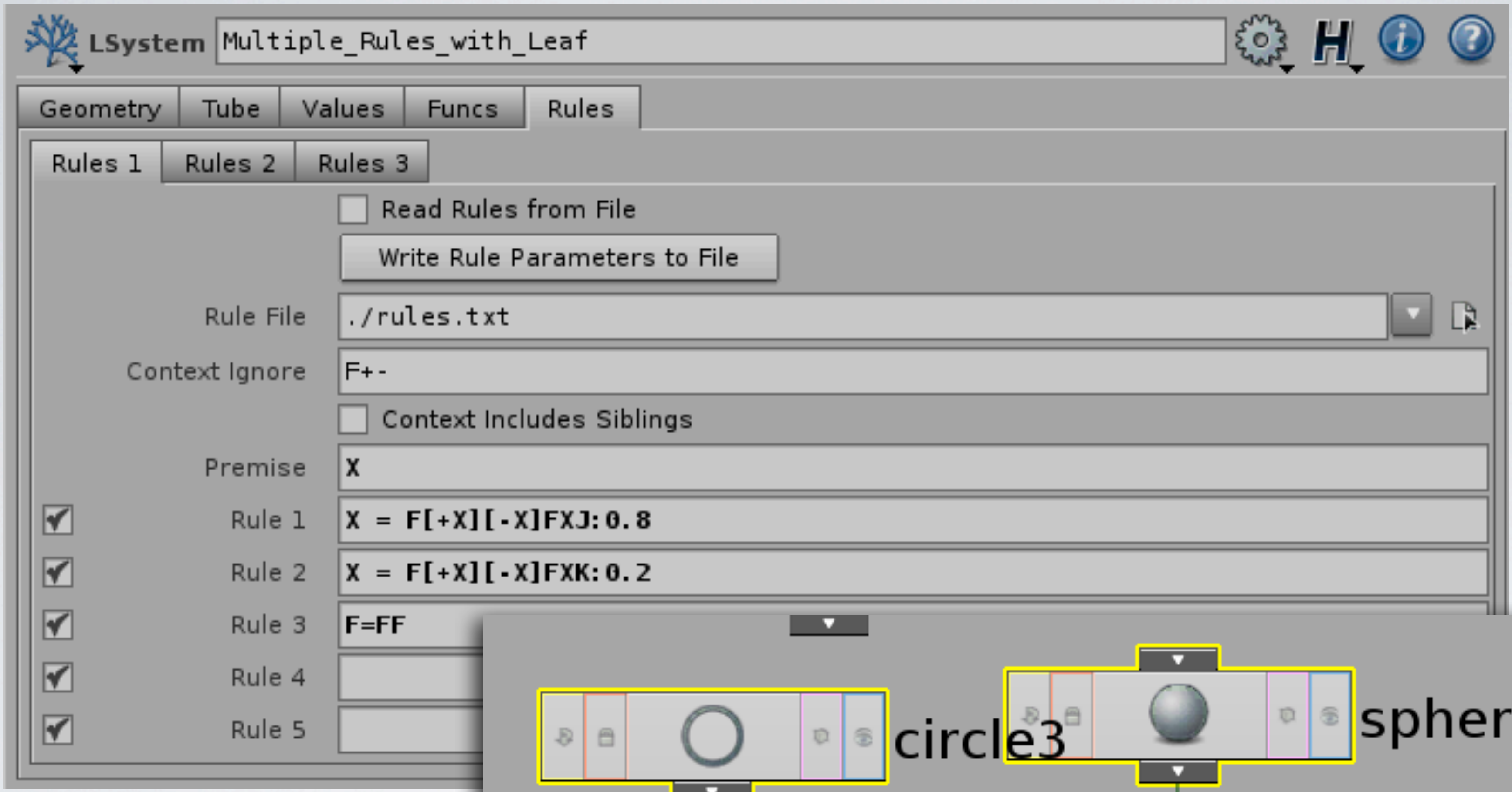
- Premise: FX
- R1: X= !/(137.5)~F[+FJ]X:0.5
- R2: X= !/(137.5)~F[+FX]X:0.2
- R3: X= !/(137.5)~F[+FK]X:0.1
- R4: X= !/(137.5)~F[+FJ]:0.2

Look like Spaghetti?

Read as the thickness of the branch decreases from generation to generation while the limbs roll 137.5 degrees and then turn randomly. R1 has a 50 percent chance of being executed. R2, 20 percent, R3, 10 percent, R4, 10 percent.

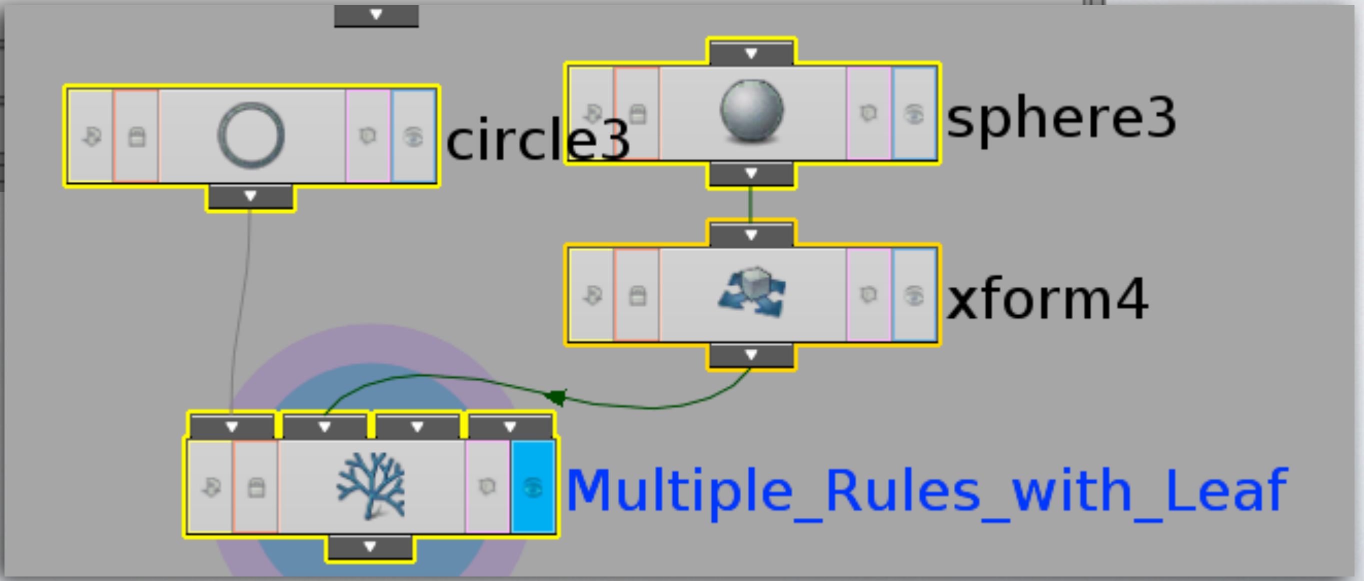
*Note: One **ugly plant** - for demo purpose only*

2D LEAF SYSTEM EXAMPLE



LSystem Multiple_Rules_with_Leaf

Rule File	./rules.txt
Context Ignore	F+-
	☐ Context Includes Siblings
Premise	X
✓ Rule 1	X = F[+X][-X]FXJ:0.8
✓ Rule 2	X = F[+X][-X]FXK:0.2
✓ Rule 3	F=FF
✓ Rule 4	
✓ Rule 5	

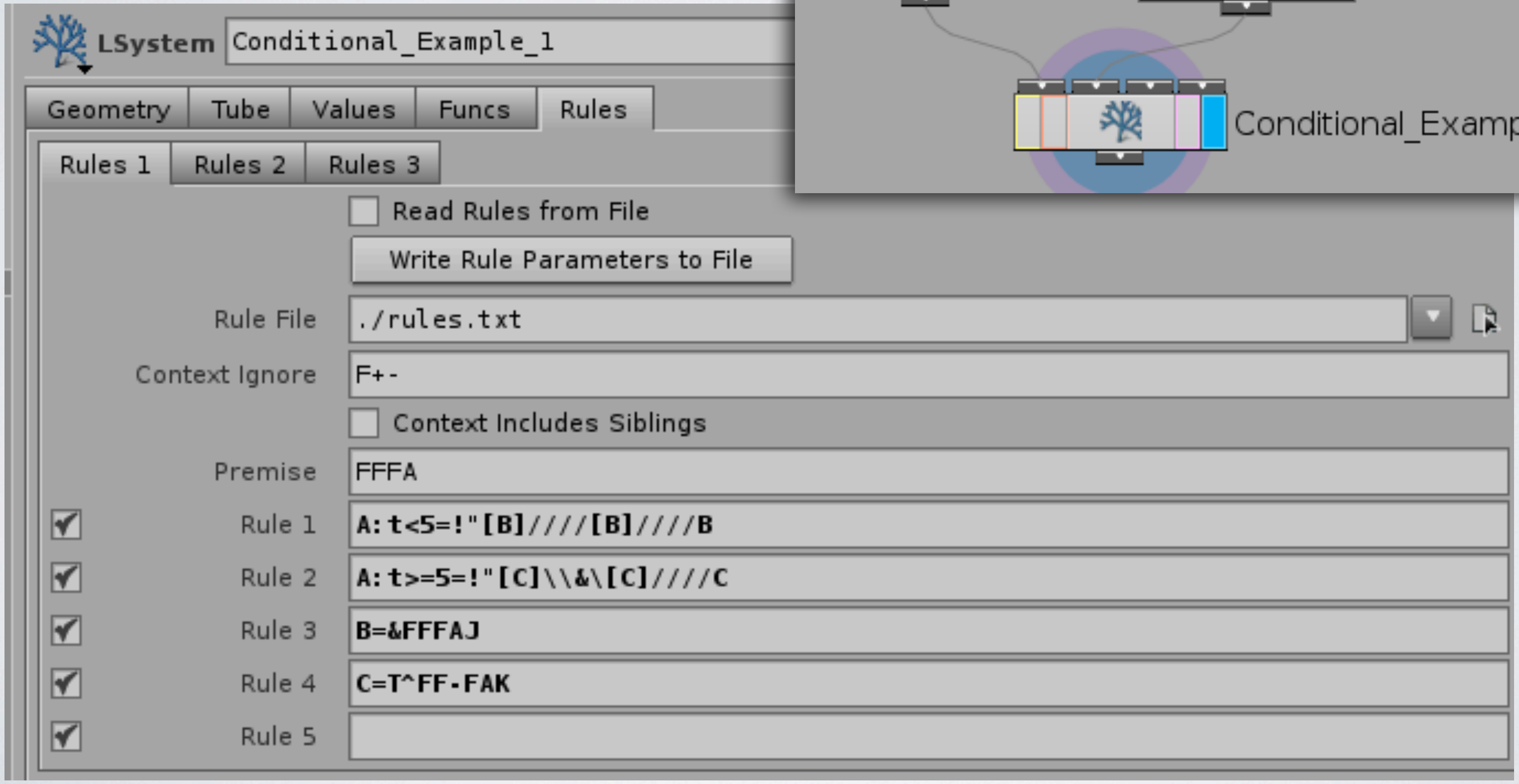


CONDITIONALS

- Condition based on generations
  - F: t<4 = F+F[F][-F]
  - F: t>4 = F++[-F]F[F][-F]
- Logical operators use the unary operators
  - F: (t<4)|(t>8) =
  - F:(t>4) &(c > 5) =

"t" stands for iteration. How many generations.

CONDITIONAL EXAMPLE



LSystem Conditional_Example_1

☐ Read Rules from File

Rule File: ./rules.txt

Context Ignore: F+-

☐ Context Includes Siblings

Premise: FFFA

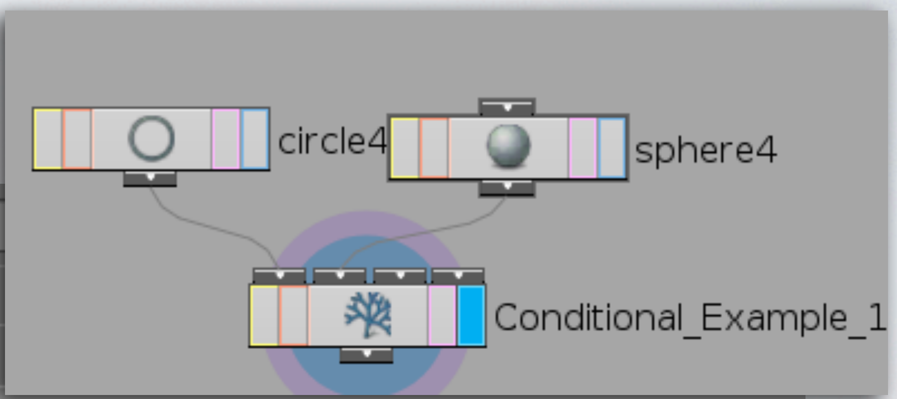
✓ Rule 1: A:t<5=!"[B]////[B]////B

✓ Rule 2: A:t>=5=!"[C]\\&\[C]////C

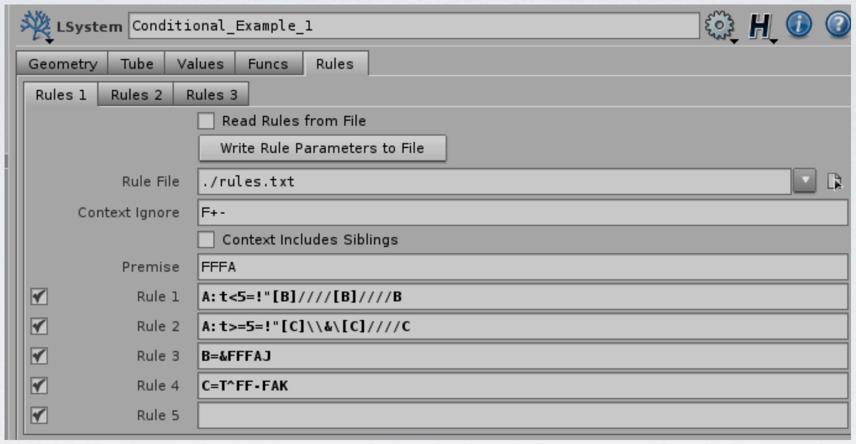
✓ Rule 3: B=&FFFAJ

✓ Rule 4: C=T^FF-FAK

✓ Rule 5:



CAN YOU READ IT?



Node name: Conditional_Example_1

- [ ] Read Rules from File
- Rule File: ./rules.txt
- Context Ignore: F+-
- [ ] Context Includes Siblings
- Premise: FFFA
- [✓] Rule 1: A:t<5=!"[B]////[B]////B
- [✓] Rule 2: A:t>=5=!"[C]\\&\[C]////C
- [✓] Rule 3: B=&FFFAJ
- [✓] Rule 4: C=T^FF-FAK
- [✓] Rule 5:

Rule 1 - If the number of generations is less then 5 Reduce the thickness of the branch and decrease the length of the branch. Then branch of B roll counter clockwise 4 times and create another branch. Roll another 4 times and append B

Your turn to do the other rules

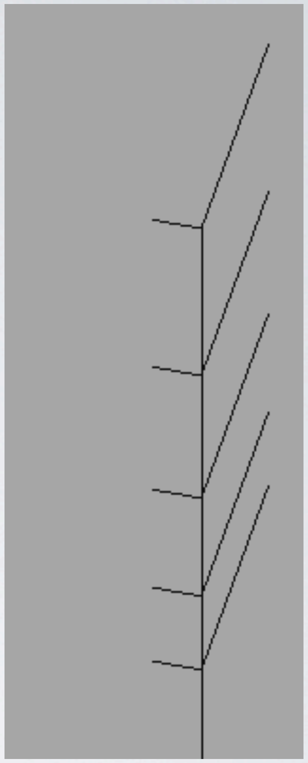
MULTIPLE SOLUTIONS FOR TURNING

- When multiple solutions present itself try to optimize for planar turning.
- Only use roll or twisting as needed

THE WHOLE TRUTH

- F(l,w,s,d)
  - l = distance
  - w = width
  - s = cross sections
  - d = divisions

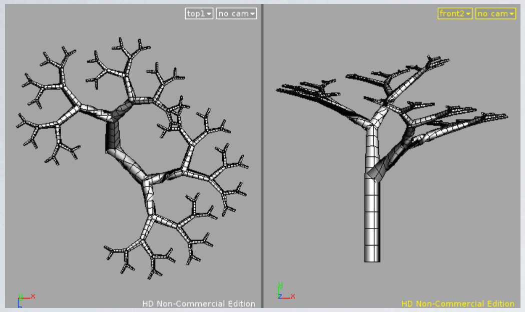
BASED ON MULTIPLE ARGUMENTS

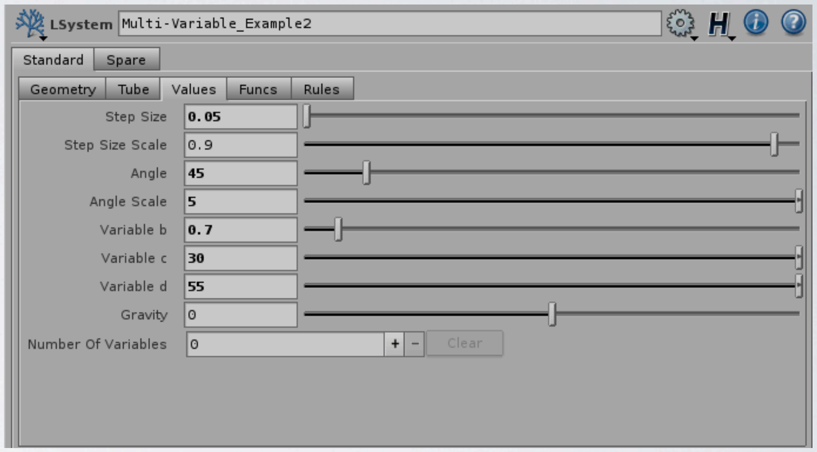


For asymmetric growth or rotation use (n) after "F" or "+ "

- Example 1
  - Premise: FX(0.1)
  - Rule 1: X(h) = F(h)[+(20)F(0.4)][-(80)F(0.1)]X(h+0.05)

BASED ON MULTIPLE ARGUMENTS



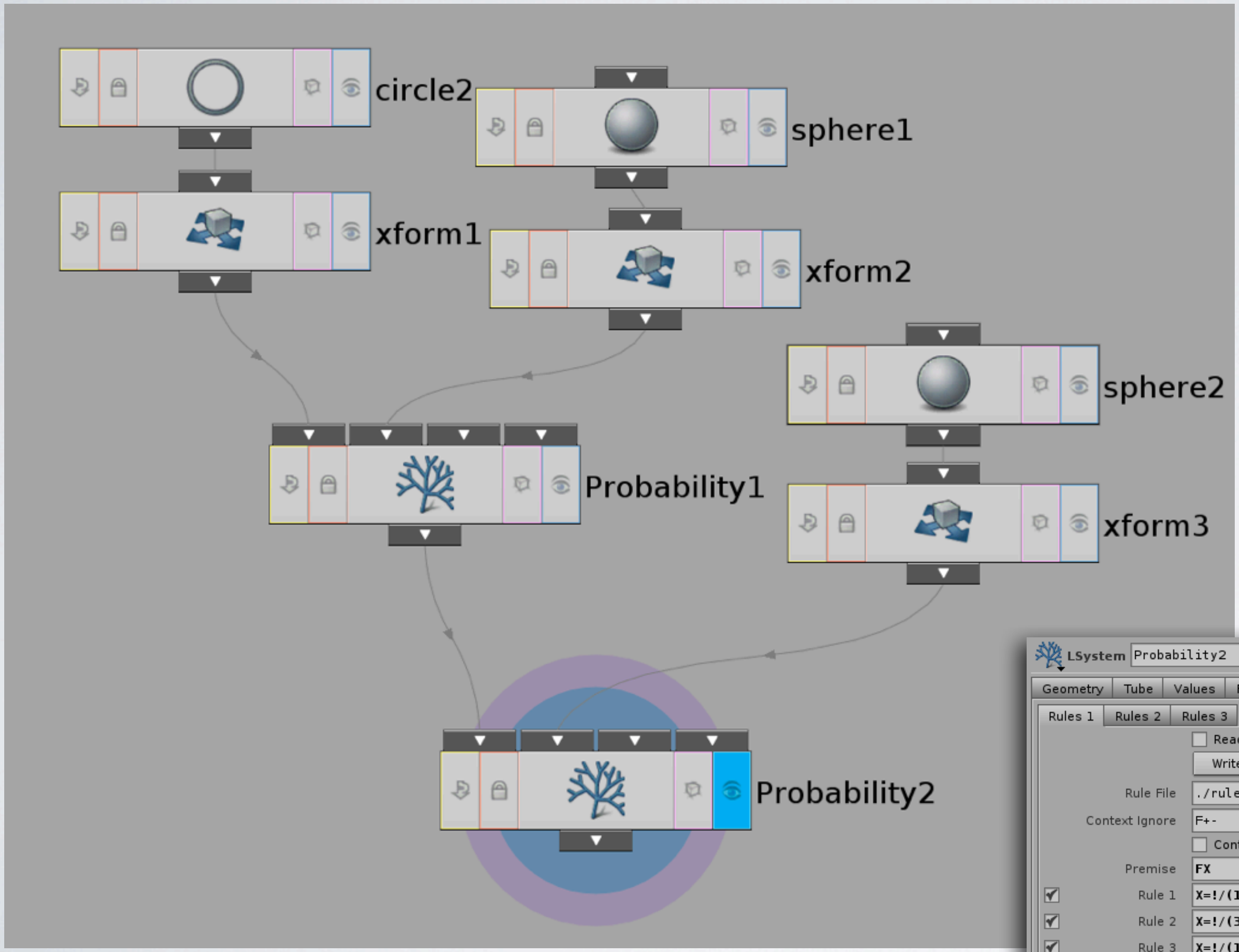


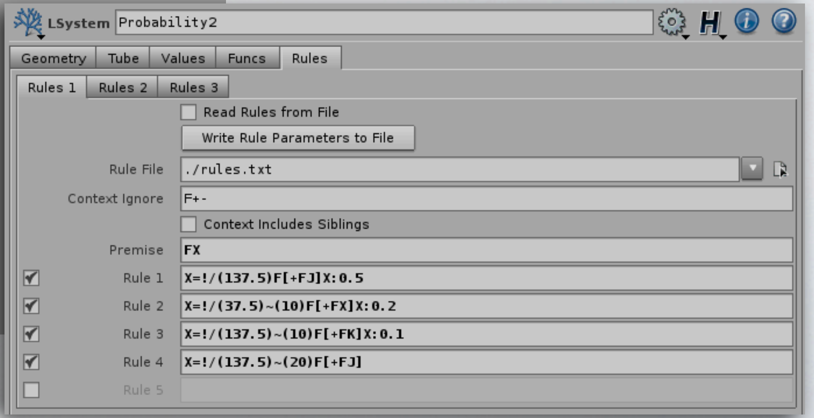
LSystem Multi-Variable_Example2

Step Size	0.05
Step Size Scale	0.9
Angle	45
Angle Scale	5
Variable b	0.7
Variable c	30
Variable d	55
Gravity	0
Number Of Variables	0

- Example 2
  - Premise: A(l,10)
  - Rule 1: A(l,w)=F(l,w)[&(c)B(l*b,w*0.707)]//(180)[&(d)B(l*0.7,w*0.707)
  - Rule 2: B(l,w)=F(l,w)[+(c)$B(l*b,w*0.707)][-(d)$B(l*0.7,w*0.707)]

SUB-SYSTEM EXAMPLE





LSystem Probability2

		[ ] Read Rules from File
	Rule File	./rules.txt
	Context Ignore	F+-
		[ ] Context Includes Siblings
	Premise	FX
✓	Rule 1	X=!/(137.5)F[+FJ]X:0.5
✓	Rule 2	X=!/(37.5)~(10)F[+FX]X:0.2
✓	Rule 3	X=!/(137.5)~(10)F[+FK]X:0.1
✓	Rule 4	X=!/(137.5)~(20)F[+FJ]
[ ]	Rule 5	

LIST OF COMMANDS

- F Turtle **Forward** one step and draw line
- f - Turtle forward without drawing
- +, - Rotate **Right**, Rotate **Left**
- J, K,M - Input from sub-context 1,2, & 3. Connection for leaves
- T(g) - Apply tropism vector (gravity
- [ ] Push, Pop Turtle State (create branch)
- /,\ Roll clockwise, counter-clockwise
- &,^ Pitch up, down
- ~ Pitch, roll, rotate random degrees
- “ ! ; multiply thickness, angle

LIST OF LOCAL VARIABLES

a
The value of the **Angle** parameter.

b
The value of the **b** parameter.

c
The value of the **c** parameter.

d
The value of the **d** parameter.

g
The age of the current rule, initially 0.

i
The offset into the current L-system string where the rule is being applied.

t
The iteration count, initially 0.

x, y, z
Current turtle position in space.

A
Arclength from the root of the tree to the current point.

L
Current length increment at the point.

T
The value of the **Gravity** parameter.

U
Color map U value.

V
Color map V value.

W
Width at the current point.

EXTERNAL EXPRESSIONS

- **expressions must be enclosed in back ticks (e.g.,)**
  - **F(`ch(crz)`)**
  - **No access to expressions except for subset**
    - **trig**
    - **rand**
    - **peak**