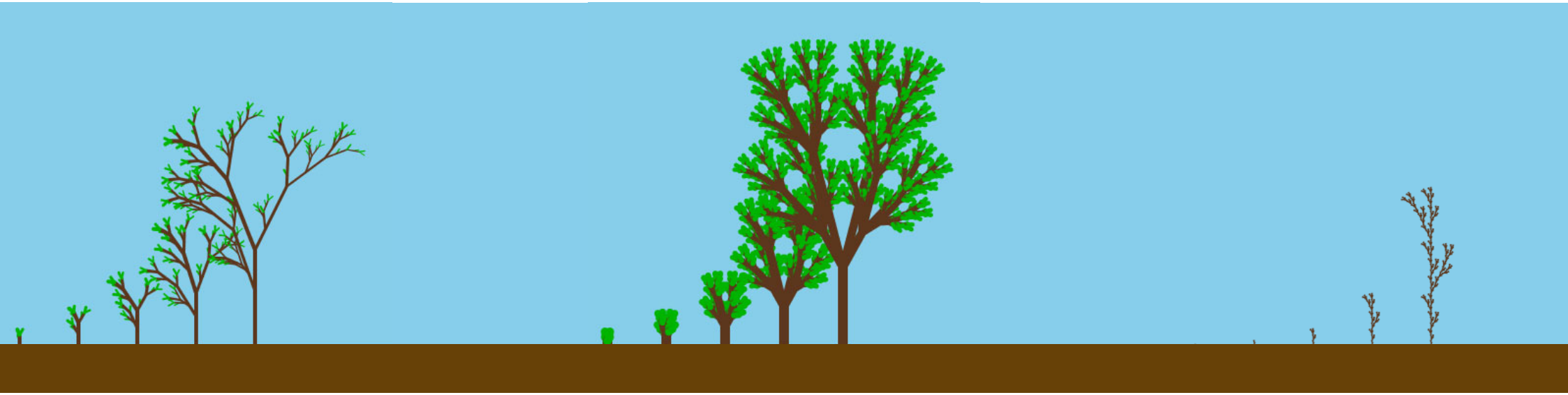
Algorithmic plants

Andrea Valente

aval@sdu.dk

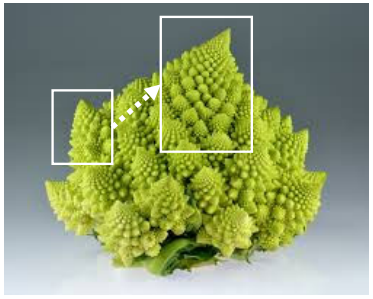
SDU Kolding

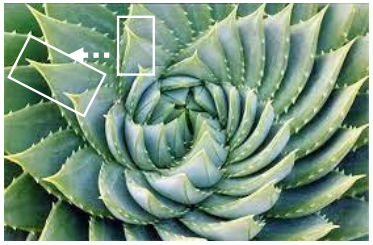


How do natural plant grow?

- *(and perhaps, can we simulate and draw them with a program?)*
- Plants are often **self-similar:**

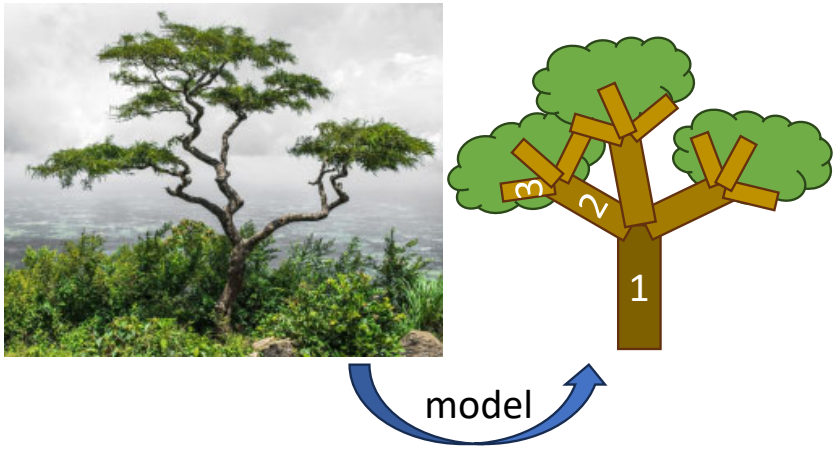
  *e.g. a **fern** is made of **smaller ferns**, …*

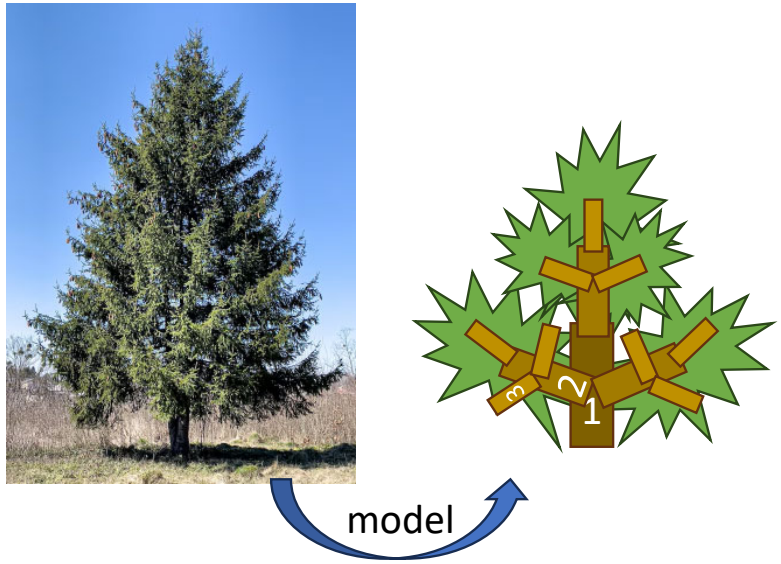




From **plant** to **model**

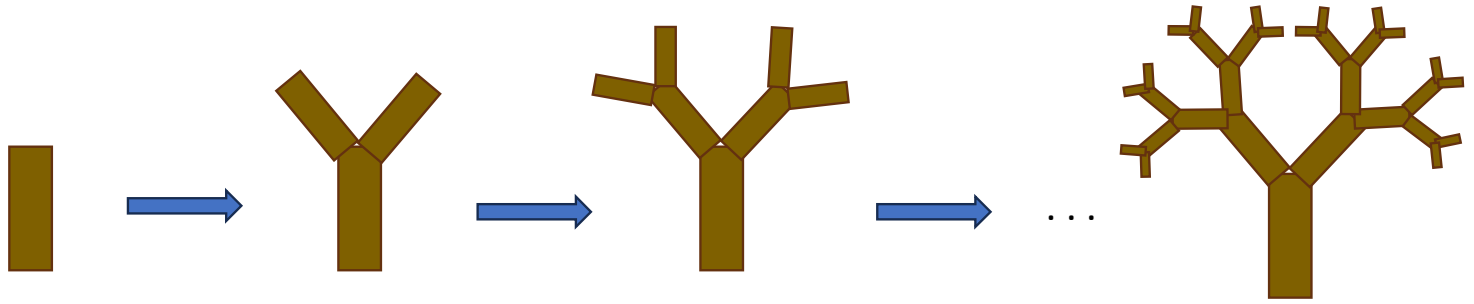
- We can use self-similarity to “simplify” the structure of a tree or plant:
  - we get a *model*
  - similar trees might have the same simplified model,
  - different trees will have different models





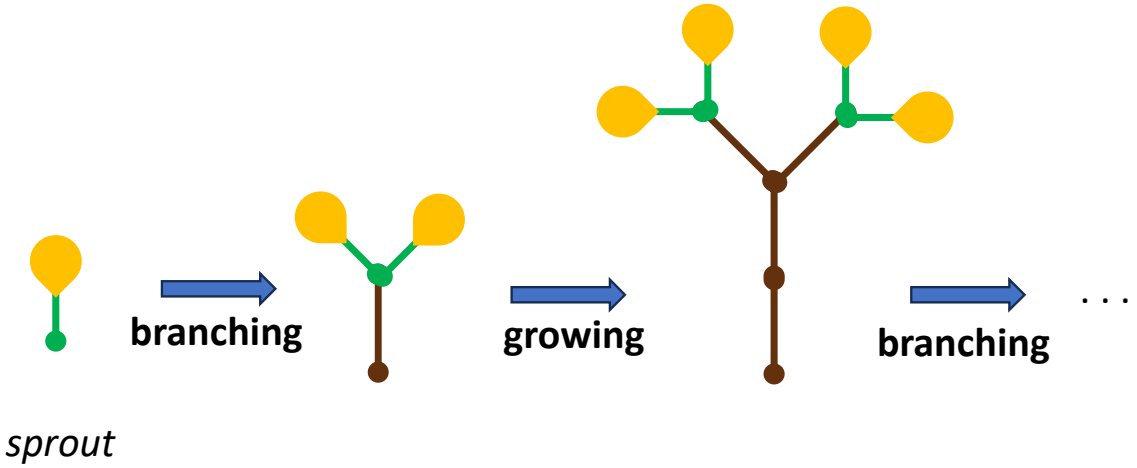
And what about *time*? Growth?

- We can define a model of a simple plant, and make it grow in steps



Growing in 2 phases

- There seem to be 2 phases in the growth of a plant:
  1. Getting taller and/or larger, i.e. growing
  2. Creating more branches, i.e. branching (still keeping *self-similarity*)



This could get a bit **repetitive**…

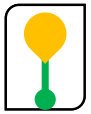
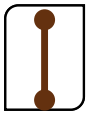
If we had **a more precise way** to describe and draw these plants, a **program** could do it for us and fast!!

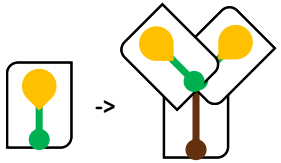
Enter: L-systems

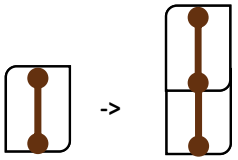
- **Lindenmayer systems (or L-systems)** were conceived as a **mathematical theory of plant development**
  - L-systems were introduced and developed in 1968 by **Aristid Lindenmayer**, a Hungarian theoretical biologist and botanist at the University of Utrecht.
  - Lindenmayer used L-systems to describe the behaviour of plant cells and to model the growth processes of plant development
- L-systems have also been used to model the **morphology** of a variety of organisms and can be used to generate self-similar fractals
  - **Morphology:** the study of the form and structure of organisms and their specific structural features

- Sources:
  - http://algorithmicbotany.org/papers/#abop
  - https://en.wikipedia.org/wiki/L-system
  - https://en.wikipedia.org/wiki/Morphology_(biology)

An L-systems is defined by **symbols** and **rules**

- Imagine that our plan is created by using cards (AKA **symbols**):
  - a **sprout** card  , its symbol could be **X**
  - and a **trunk** card  and its symbol can be **F**
- And some **rules** to replace (AKA rewrite) the cards:
  - start with X
  - X *-becomes->* F[+X][-X]
  - F *-becomes->* FF





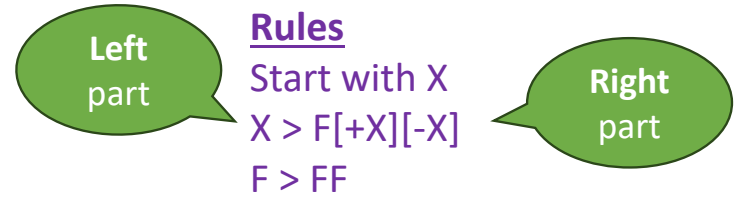
Rules expresses visually

Growing the plant = *rewrite* the symbols

- The L-system **separates** the DESCRIPTION of a plant from how it LOOKS
  - we rewrite the symbols to simulate the plant's growth,
  - then we draw the plant, from the symbols

- Rewriting works this way:
  - **Start** with a symbol X
  - Look at every symbol:
    if it **matches** the **left-part** of a rule,
    then **change** that symbol with the **right-part** of the rule
  - Then **repeat** the last step, until you are happy whit your plant! ;)
- Example:

  X => F[+X][-X] => FF[+F[+X][-X]][-F[+X][-X]] => . . .



Growing the plant *(step by step)*

- Rewriting works this way:
  - **Start** with a symbol X
  - Look at every symbol:
    if it **matches** the **left-part** of a rule,
    then **change** that symbol with the **right-part** of the rule
  - Then **repeat** the last step, until you are happy whit your plant! ;)
- Example:

  X => F[+X][-X]

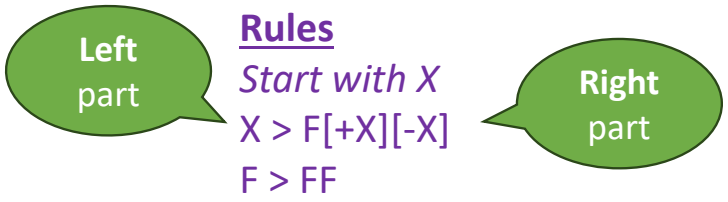
  then:

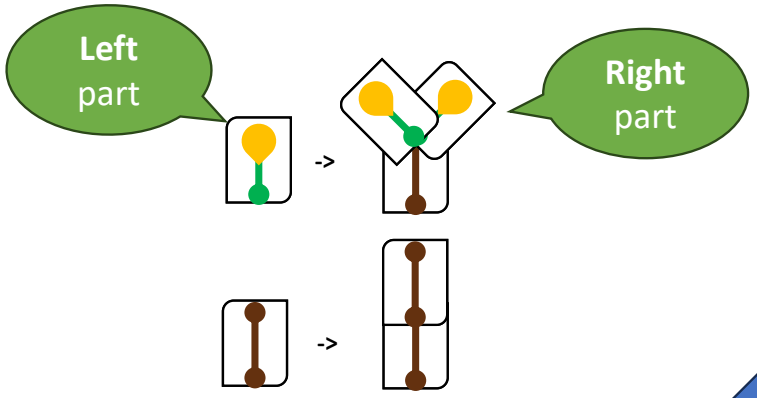
F[+X][-X] => FF[+F[+X][-X]][-F[+X][-X]] => . . .

FF

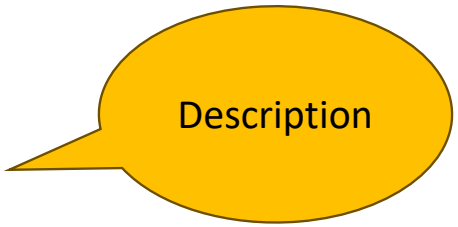
F[+X][-X]

F[+X][-X]





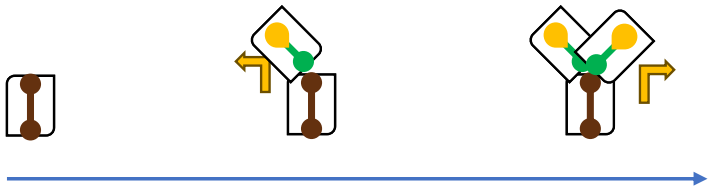
How to draw a plant, from its description

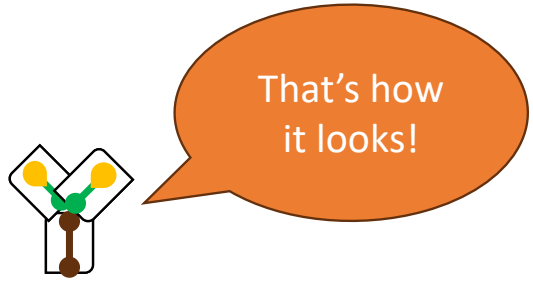


- We start from the text:

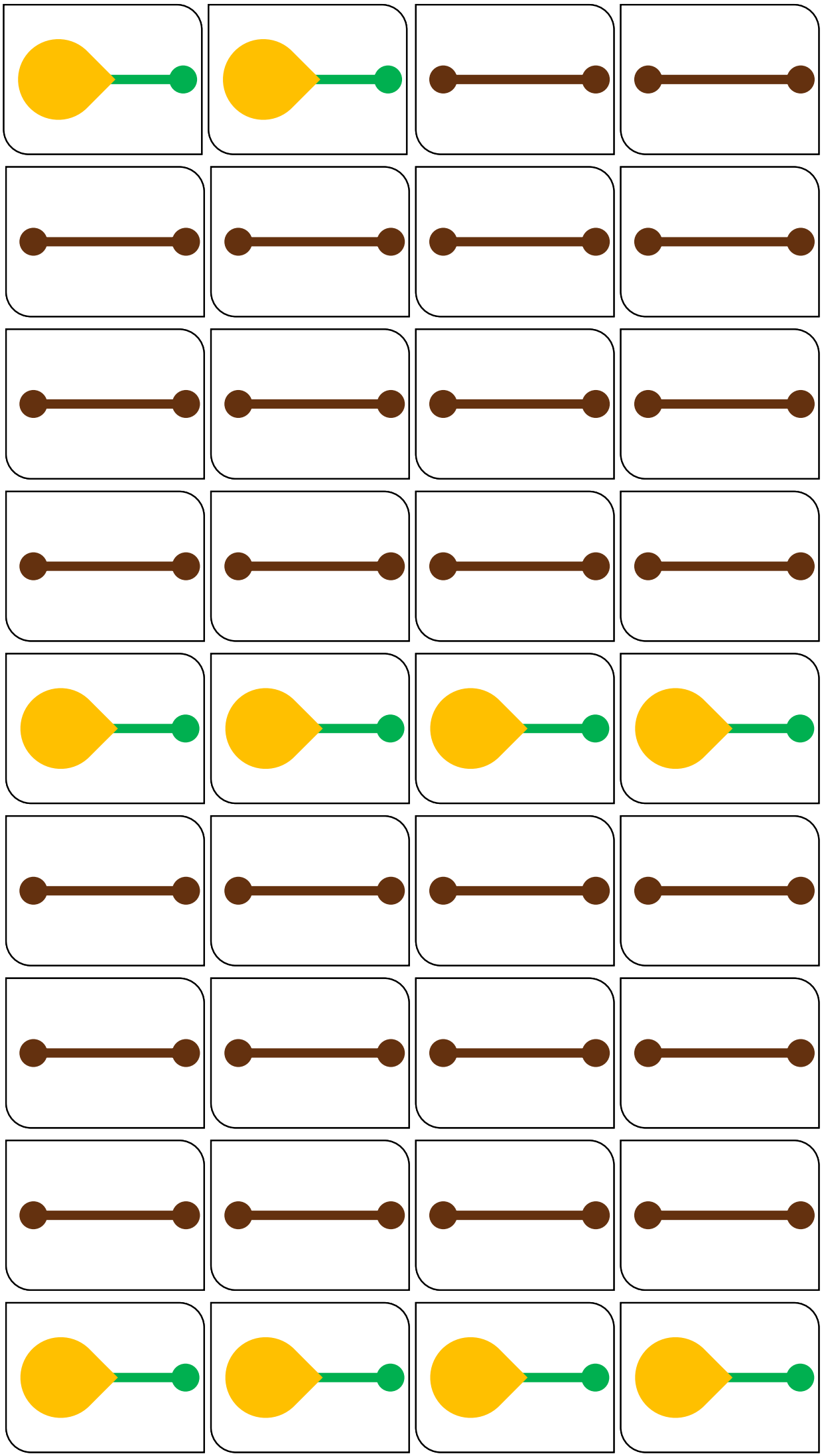
  X => F[+X][-X] => FF[+F[+X][-X]][-F[+X][-X]] => . . .

- And interpret:
  - **X** as a **sprout** card 
  - **F** as a **trunk** card 
  - **+** as a turn of 45 degrees  , **-** as a counter-clock turn of 45 degrees 
  - **[** as start a branch, and **]** as start go back to the last branch
- Example: let's **draw** F[+X][-X]





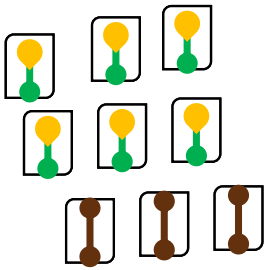
print me :) print me :) print me :)



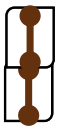
Let’s try together…

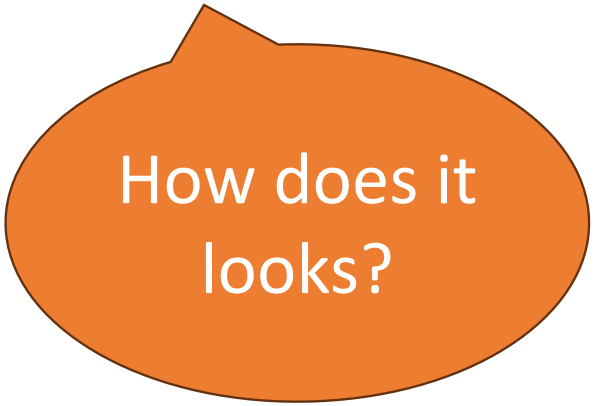
How does it looks?

- Let’s draw this text plant: FF[+F[+X][-X]][-F[+X][-X]]
  - **X** as a **sprout** card 
  - **F** as a **trunk** card 
  - **+** as a turn of 45 degrees , **-** as a counter-clock turn of 45 degrees
  - **[** as start a branch, and **]** as start go back to the last branch



?




How does it looks?

Description

X

⇓

F[+X][-X]

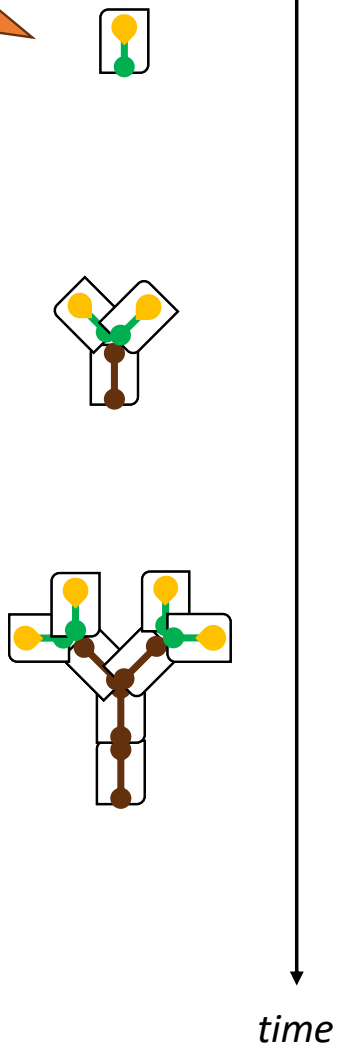
⇓

FF[+F[+X][-X]][-F[+X][-X]]

⇓

... a very long text ...

That's how it looks!



time

Alternative way to draw a text plant

- Use a **pen**! :D

X

⇓

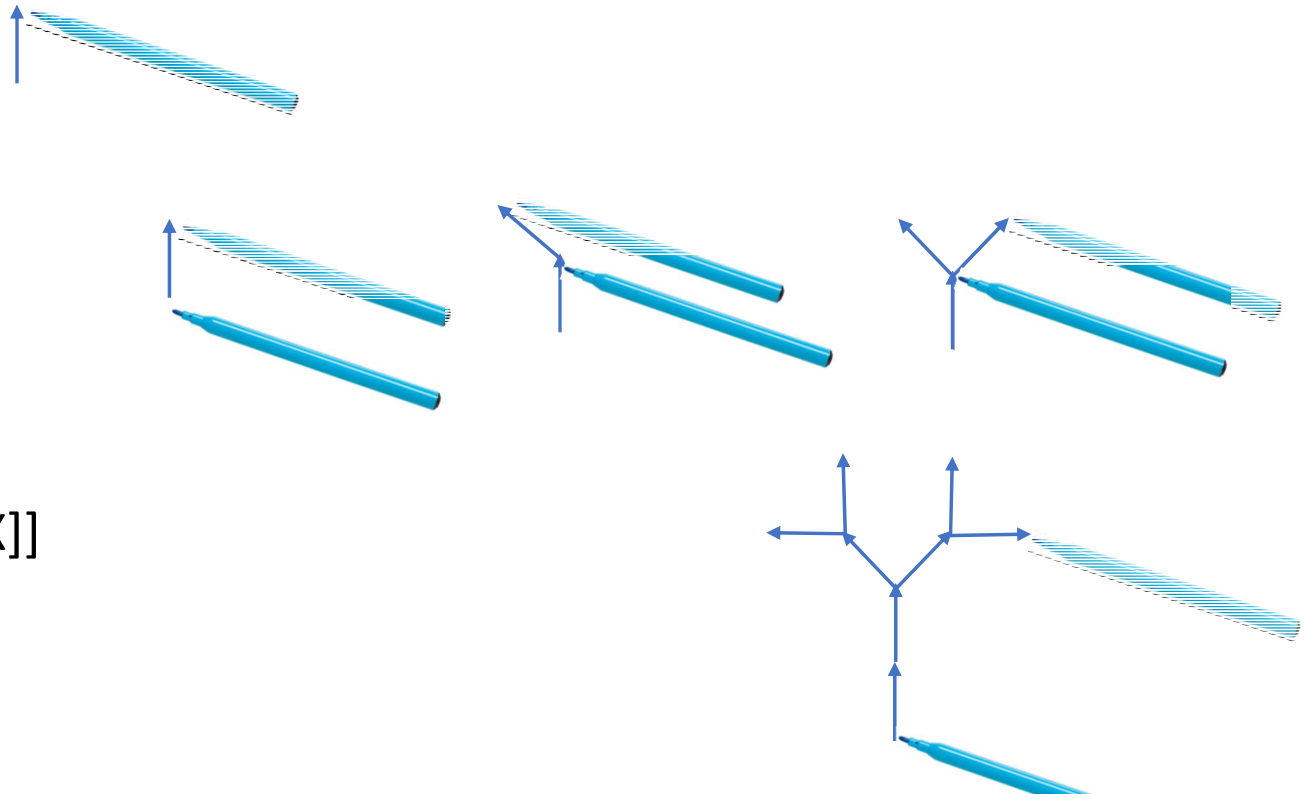
F[+X][-X]

⇓

FF[+F[+X][-X]][-F[+X][-X]]

⇓

…



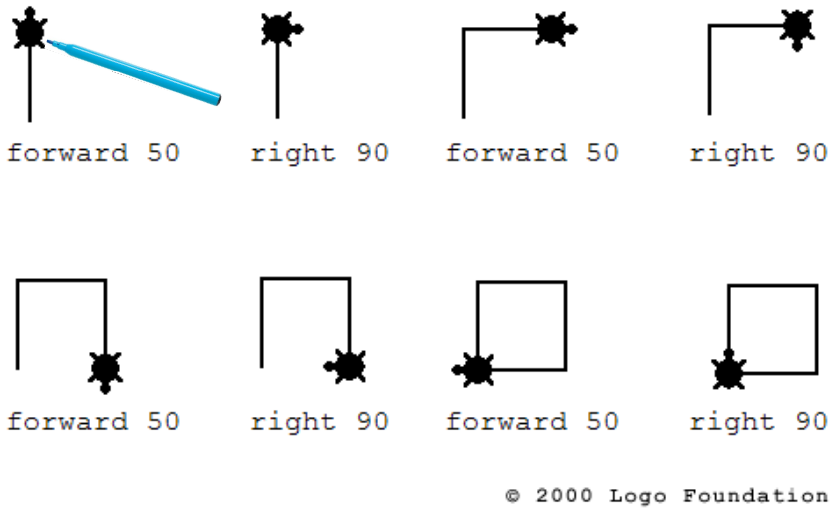
LOGO – a turtle with a pen

Could I instruct the computer draw the tree for me?

- **LOGO programming language**
  - You control a **turtle** on screen
  - **It has a pen**, can lift it or put it down
  - The commands are:

    Forward 10 steps or F 10
    Rotate Right 90 degrees or R 90
    Rotate Left 45 degrees or L 45

  - Example of shapes you can draw using LOGO

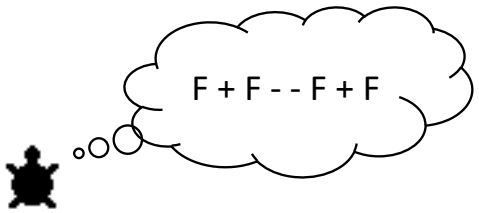


LOGO with simpler instructions

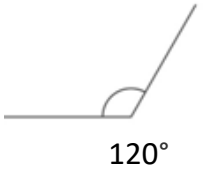
- We can simplify LOGO instructions even more:
  - We can decide that we work with fixed **length** of 10 steps and fixed **angle** of 60 degrees

- And the instructions could be *super simple*:
  - **F** could be forward **length** steps (here 10)
  - **+** could be turn right **angle** degrees (in this case 60 degrees)

- What will the turtle draw with the following instructions:
  - **F + F - - F + F** ??



”Be the turtle”

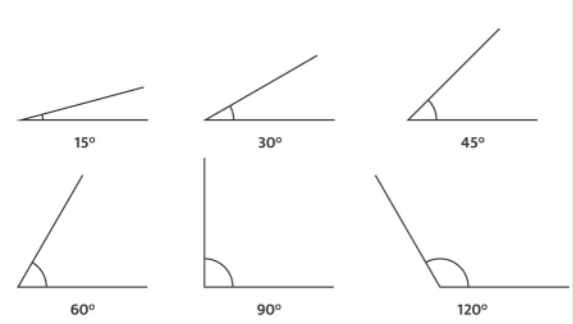


- What will the turtle draw with the following instructions:
  - **F + F - - F + F** ??
    given and angle of 120 degrees for + and -

- But if we change the **angle** to 90 degrees, and draw the same instructions?

“Be the turtle” … with this L-system

- Let’s consider a simpler L-System, with only 1 rule:
  X > FF[+X]-F
- Do this:
  1. Start with X, then rewrite 2 times (AKA grow the text plant 2 steps)
  2. Now, draw each plant description using the *LOGO turtle* method (AKA a pen).
     Let’s fix the **angle** at 30 degrees (*circa*), so **+** is 30 deg. and **–** is -30 deg.



Results (?)

- … tell me what you've got …

    1. *first the rewriting steps*

    2. *then… what does your plant look like?*

So, to create **algorithmic plants**, we just need

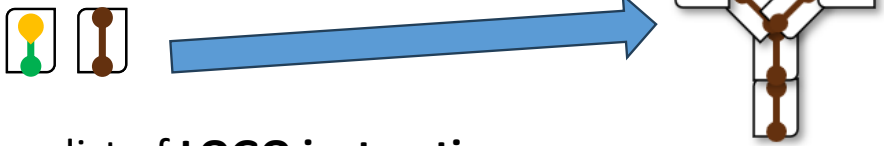
- A set of rules
  - E.g.

start with X
X > F[+X][-X]
F > FF

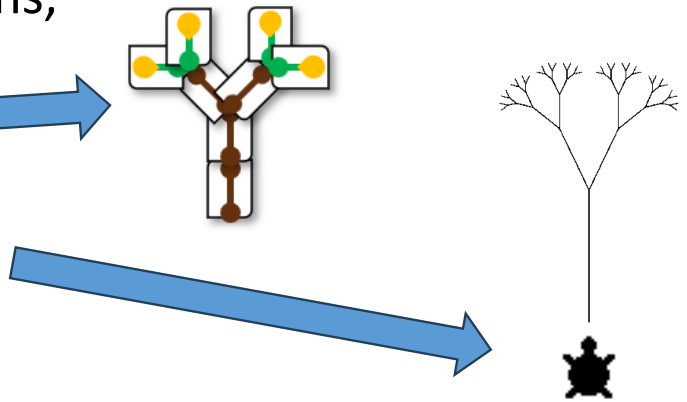
  - then we just **match and rewrite** using the rules

X => F[+X][-X] => FF[+F[+X][-X]][-F[+X][-X]] => . . .

- A way to draw the text plants
  - we need to decide about a fixed **angle** for the rotations,
  - then:
    - we can **use our cards**



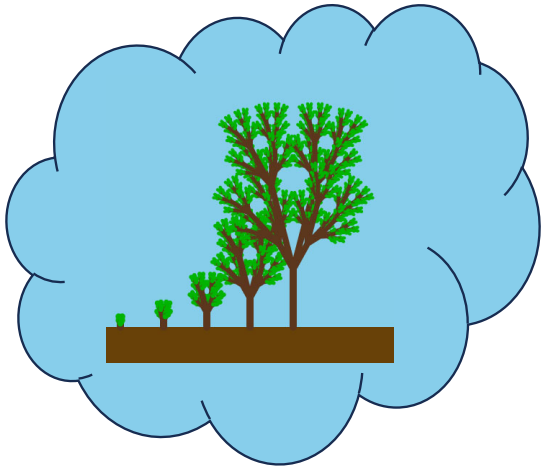
    - or we can read the text plant as a list of **LOGO instructions**



A program to play with L-systems

I would like a program that:

1. Let's me define my L-System (AKA write some rules)
2. Then it automatically:
   - Rewrites a few times, to get a sequence of plant descriptions
   - And for each, use a LOGO turtle to draw the look of a growing plant



Program in p5.js

Scan me!

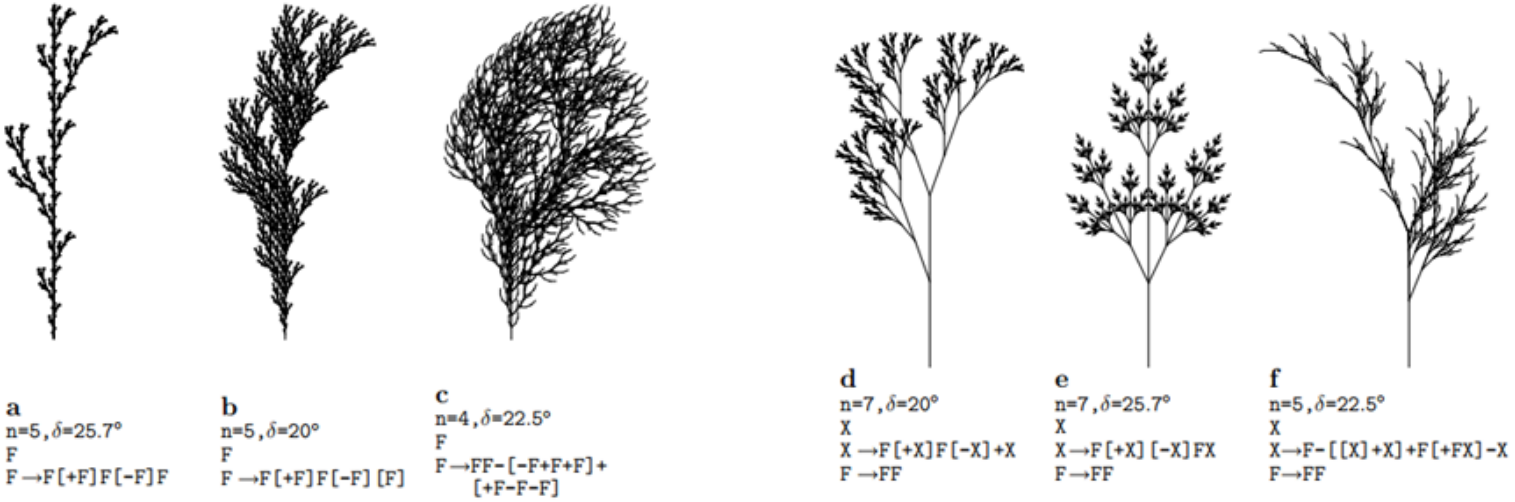
- In your browser, open https://editor.p5js.org/andrea270872/full/3Z0R3PxZW
- Let's try out:
  - **System 1:**
    X>F
    F>F[-F]F[+F]F
    Set the "angle" to 25 and "forward" to 3, then play with the params to create *alternative looks*
  - **System 2:**
    X>F
    F>FF+[+F-F-F]-[-F+F+F]
    Set the "angle" to 22.5 and "forward" to 4, then play with the parameters
  - **System 3:**
    F>FF
    X>F[-X][+X]
    Set the "angle" to 30 and "forward" to 9, then play with the parameters.
    Use the "save snapshot" button, to create an "animation" of your plan, as the params change.

- If you are interested in the code, it's Javascript with the P5.js library:
  https://editor.p5js.org/andrea270872/sketches/3Z0R3PxZW

More L-Systems you might want to try…

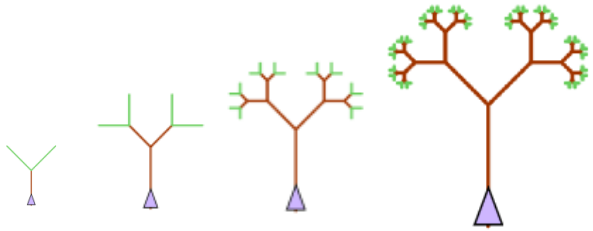
Inspiration

- They are expressed a bit differently…
- Can you find out how to write them in our program?

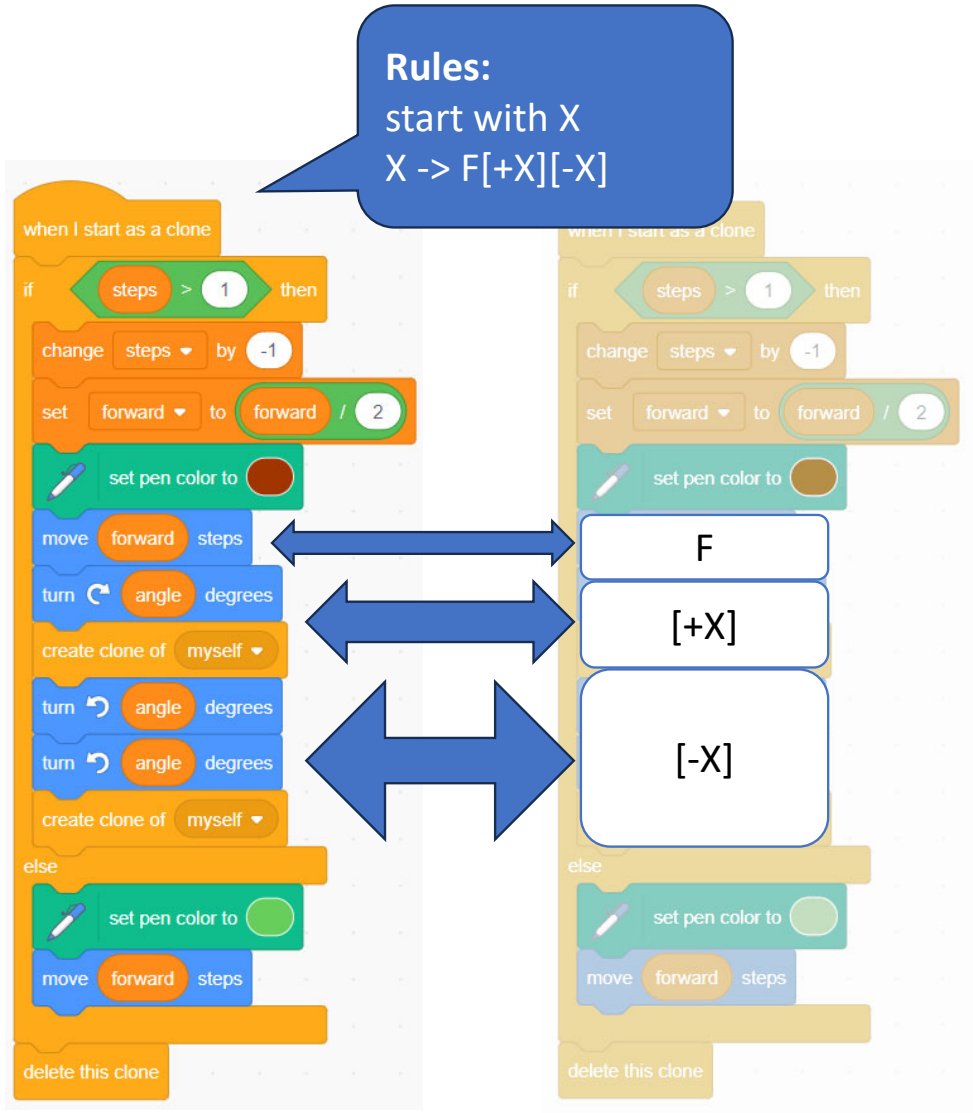


Program in **Scratch**

- In Scratch https://scratch.mit.edu/projects/969810191/editor/
- This program is simpler than the other,
  but if you know Scratch it might be more *readable*

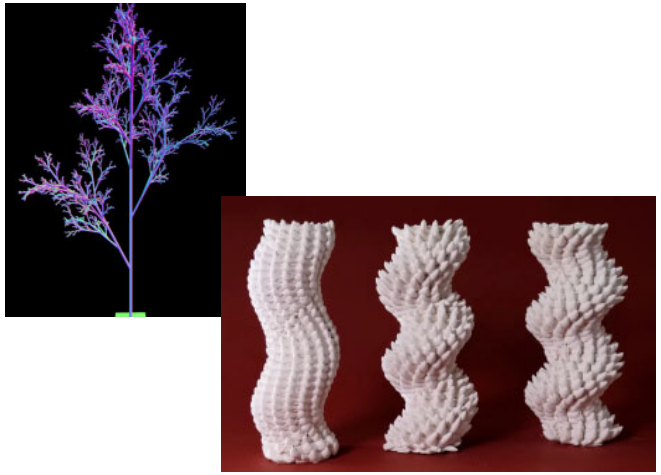


- The code does this -------->



Questions?

Other applications of L-Systems



- 3D plants:
  https://apps.simshadows.com/3d-lsystems-explorer/

- For 3D printing:
  https://coudre.studio/projects/printed-l-systems/

- Music
  (scientific paper https://www-users.york.ac.uk/~ss44/bib/ss/nonstd/eurogp05.pdf)
  https://www.youtube.com/watch?v=TgIUGcTK2FU [video]

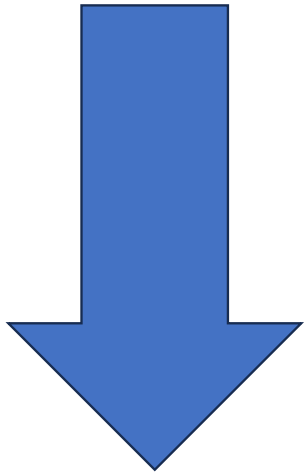
- Graphic tool Houdini
  https://www.youtube.com/watch?v=8jYNmf1VzsQ

More info

- A intro good video https://www.youtube.com/watch?v=feNVBEPXAcE
- Classic Book: https://en.wikipedia.org/wiki/The_Algorithmic_Beauty_of_Plants
  - FREE PDF: http://algorithmicbotany.org/papers/abop/abop.pdf
- Research paper (with great visual explanations and illustrations): http://algorithmicbotany.org/papers/sigcourse.2003/2-1-lsystems.pdf
- Article about L-systems https://medium.com/@hhtun21/l-systems-draw-your-first-fractals-139ed0bfcac2

Solution

“Be the turtle” … with this L-system



X => FF[+X]-F

=> FF[+FF[+X]-F]-F

=> FF[+FF[+FF[+X]-F]-F]-F

=> FF[+FF[+FF[+FF[+X]-F]-F]-F]-F

=> . . .