POLITECNICO
MILANO 1863

IMAGE AND SOUND
ISPG
PROCESSING GROUP

CREATIVE PROGRAMMING AND COMPUTING

Grammar- and Automata-based agents

MODELLING CREATIVITY - A RANDOM PROCESS

- The oldest, simplest and most common way to model **creativity** is through the introduction of uncertainty (**randomness**)

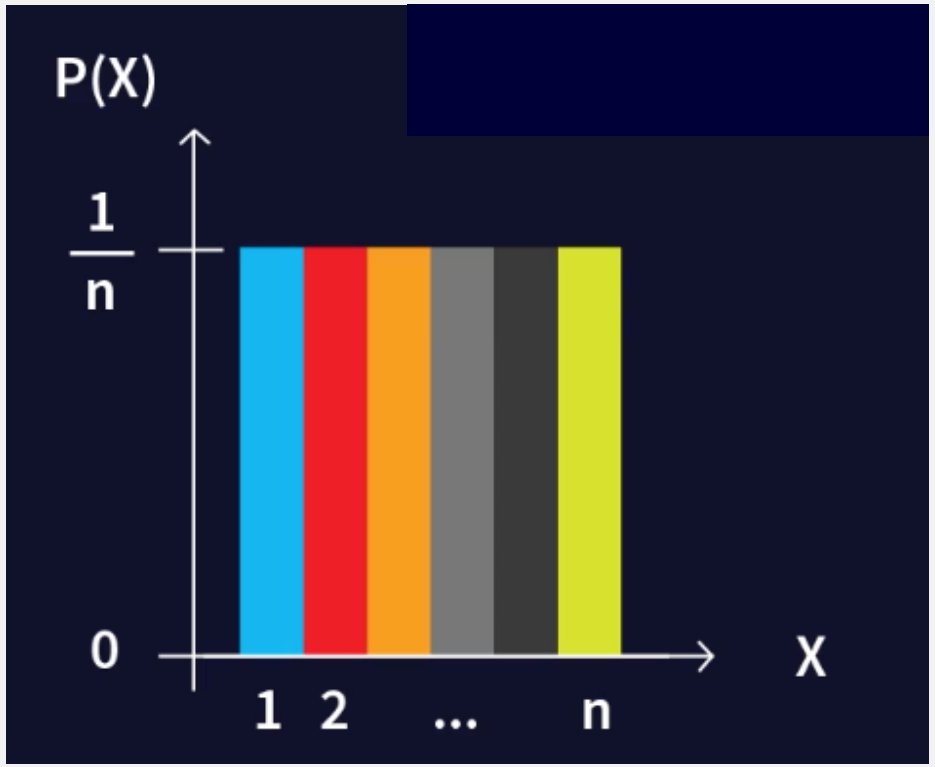
- The artists selects a space of parameters where values can take values in a certain ranges
- Value are then randomly selected

- **Definition:**
  - **Given a variable X defined over a discrete values space (the domain D), a random draw is giving equal probability to all the option from D**

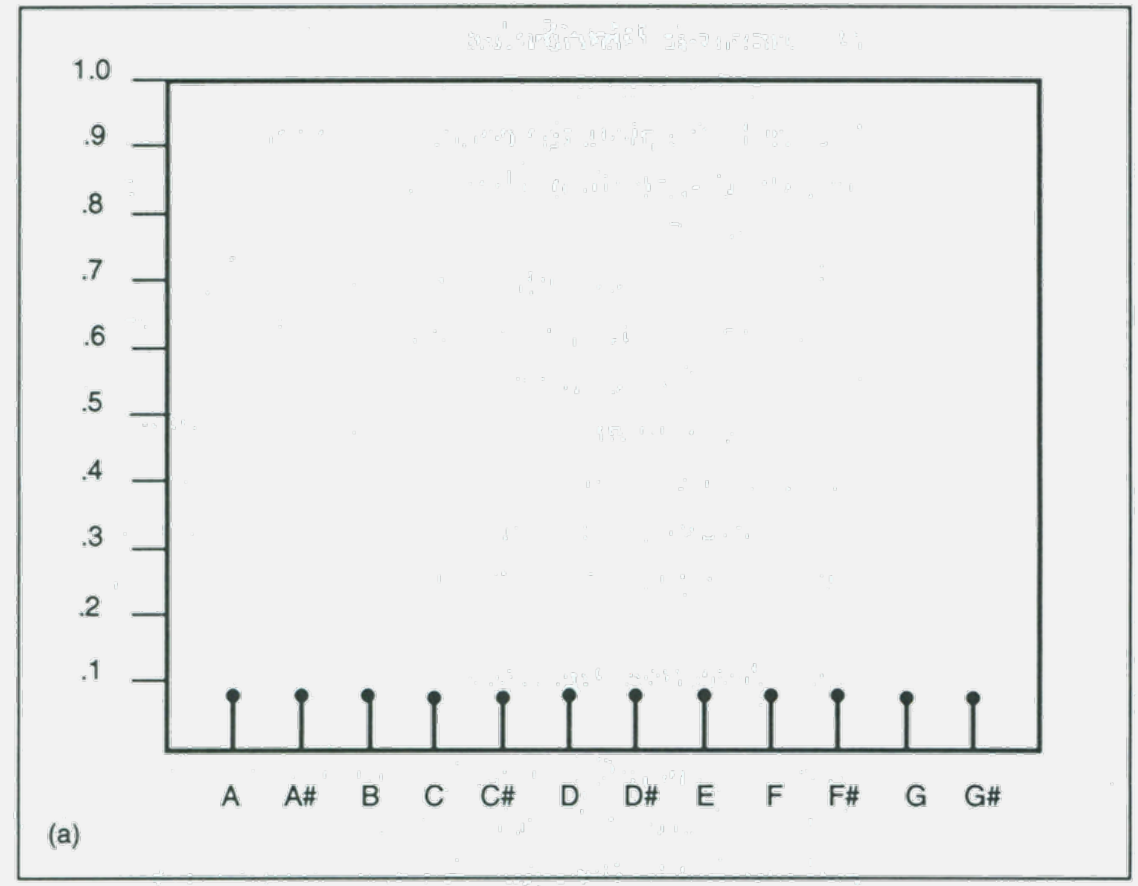
MODELLING CREATIVITY - A RANDOM PROCESS

- **Probability variable X**: variable that can assume values according to a **probability distribution**
- Probability distribution gives the likelihood **P(X)** of each possible values X can assume

  - Uniform distribution
  - It is the distribution we describe as random process



UNIFORM DISTRIBUTION IN MUSIC



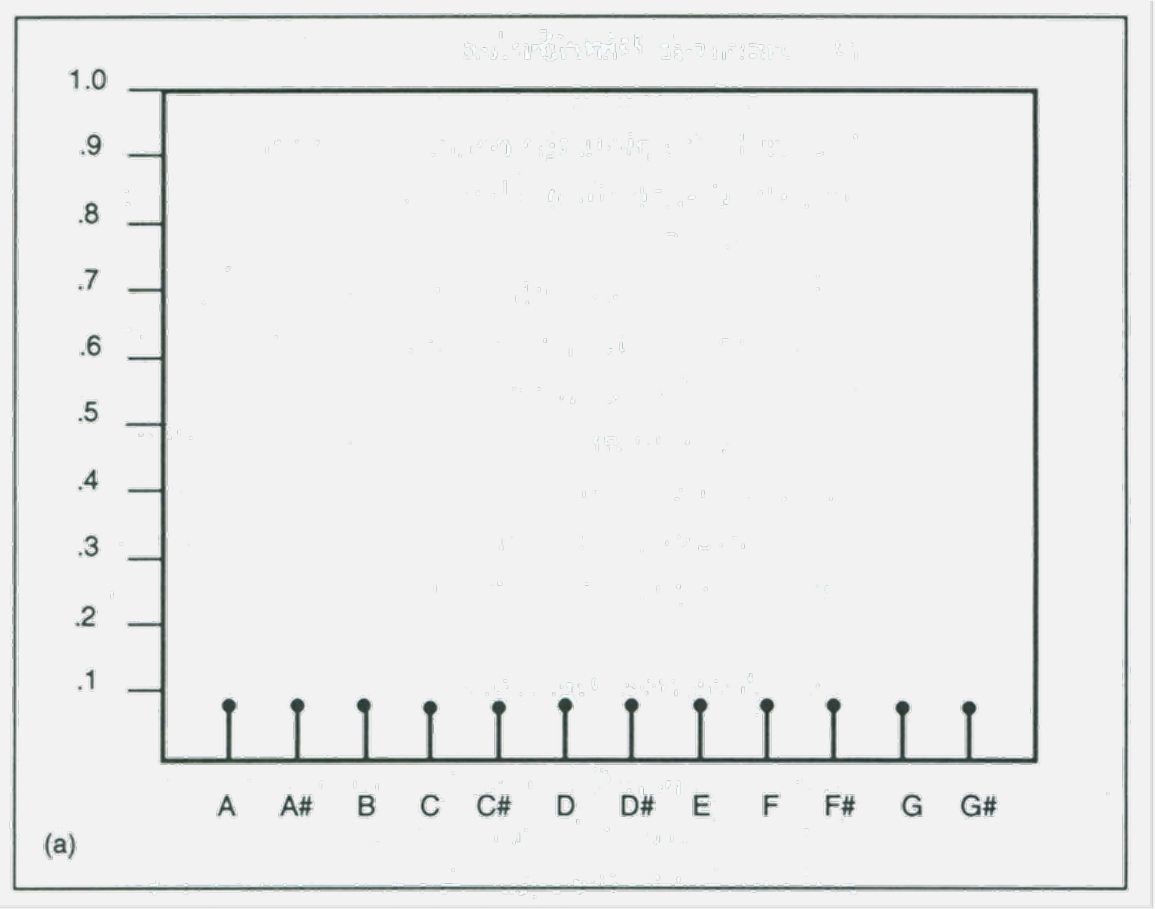
From [ROADS 96]: Each line on the vertical axis represents the probability of a corresponding value on the horizontal axis. In this case, the horizontal axis represents the pitches of a chromatic scale. Uniform distribution: every pitch has the same probability: 0.8

In SuperCollider: **Prand**, **Pxrand**

Randomness_uniform

UNIFORM DISTRIBUTION IN MUSIC

- **Uniform distribution for random walk**
- Generates a sequence of notes where each note is randomly selected



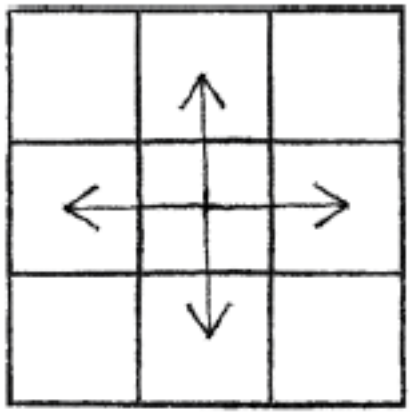
- Tare processes in which, while the next note in a sequence is random, but this value depends on the current note
- Random walk: given the current note, the value of the next note is a half-step higher or a half-step lower depending on the flip of a fair coin (uniform distribution)

In SuperCollider: **Pwalk**

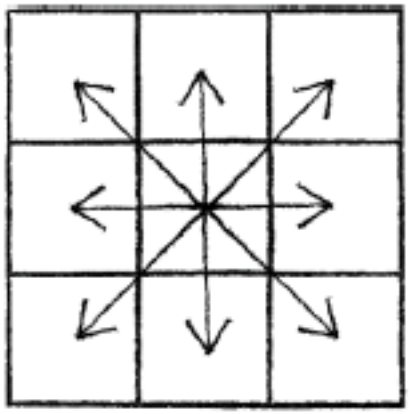
RandomWalk

UNIFORM DISTRIBUTION IN COMPUTER GRAPHICS

- **Uniform distribution for random walk**
- Generates a sequence of graphic points where the location is randomly selected
- Given the location of a point, the next point will be randomly selected according to some linking rules



4 possible steps

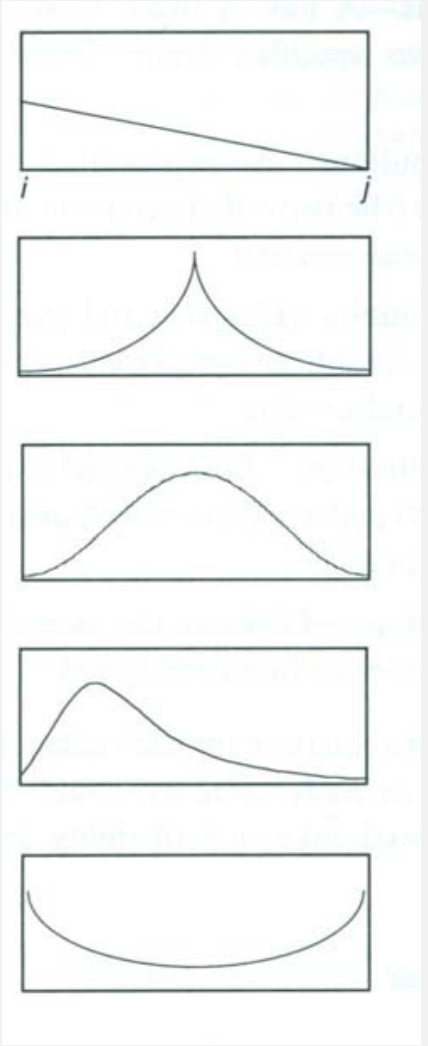


8 possible steps

RandomWalkTrail

OTHER DISTRIBUTIONS

- It is possible to have other probability distributions
  - Linear distributions
  - Exponential distributions
  - Bell shaped distribution
  - Asymmetrical distributions
  - U-shaped distributions

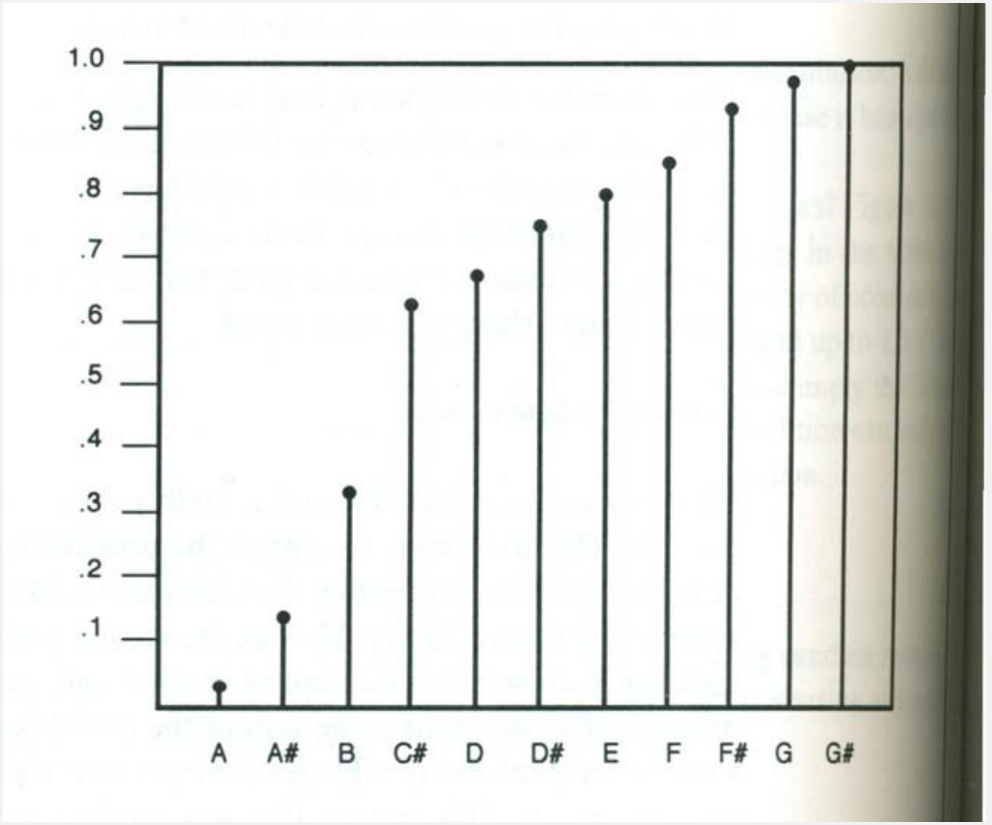


In SuperCollider: **Pwrand**

Randomness_no_uniform

CUMULATIVE DISTRIBUTION IN MUSIC

- Cumulative distribution is interesting
- 1°step: Convert the probability distribution into a cumulative distribution by summing the values successively



- 2°step: Compare the pseudorandom number with the numbers in the cumulative distribution:

```
for i=A to G#
  if random_numb is less than ptable(i)
  then
     play(i)
end
```

FURTHER CONSIDERATIONS

- There are many extensions of the straightforward probability-based walk
- Allow the possibility to change probabilities over time: the piece starts with one distribution at the beginning and with another at the end (e.g. through some sort of interpolation).
- The selection process can be replicated to other
  - musical parameters such as register, duration, dynamic intensity,....
  - graphical parameter like brightness or colour

THE LEVY FLIGHT

- Monte Carlo method
  - pick two random numbers
  - the second one is a "qualifying random value." It will asses whether to use the first one or throw it away and pick another one
- A specific case is when the random number is the probability itself

1. Pick a random number: R1

2. Compute a probability P that R1 should qualify. Let's try: P = R1.

3. Pick another random number: R2

4. If R2 is less than P, then we have found our number—R1!

5. If R2 is not less than P, go back to step 1 and start over.

THE LEVY FLIGHT

- Here we are saying that the likelihood that a random value will qualify is equal to the random number itself
- Let's say we pick 0.1 for R1. This means that R1 will have a 10% chance of qualifying. If we pick 0.83 for R1 then it will have a 83% chance of qualifying
- The higher the number, the greater the likelihood that we will actually use it
- In our example: *probability = pow(1.0 - r1,8)*
- Smaller jump are preferred

A SONIFICATION EXAMPLE THE LEVY FLIGHT

- Computer graphics random walk based on Monte Carlo probability model
- Uncertainty is not only applied to the direction of the next point but also to the distance
- Current position (x,y) sent to the Wobble Bass via OSC

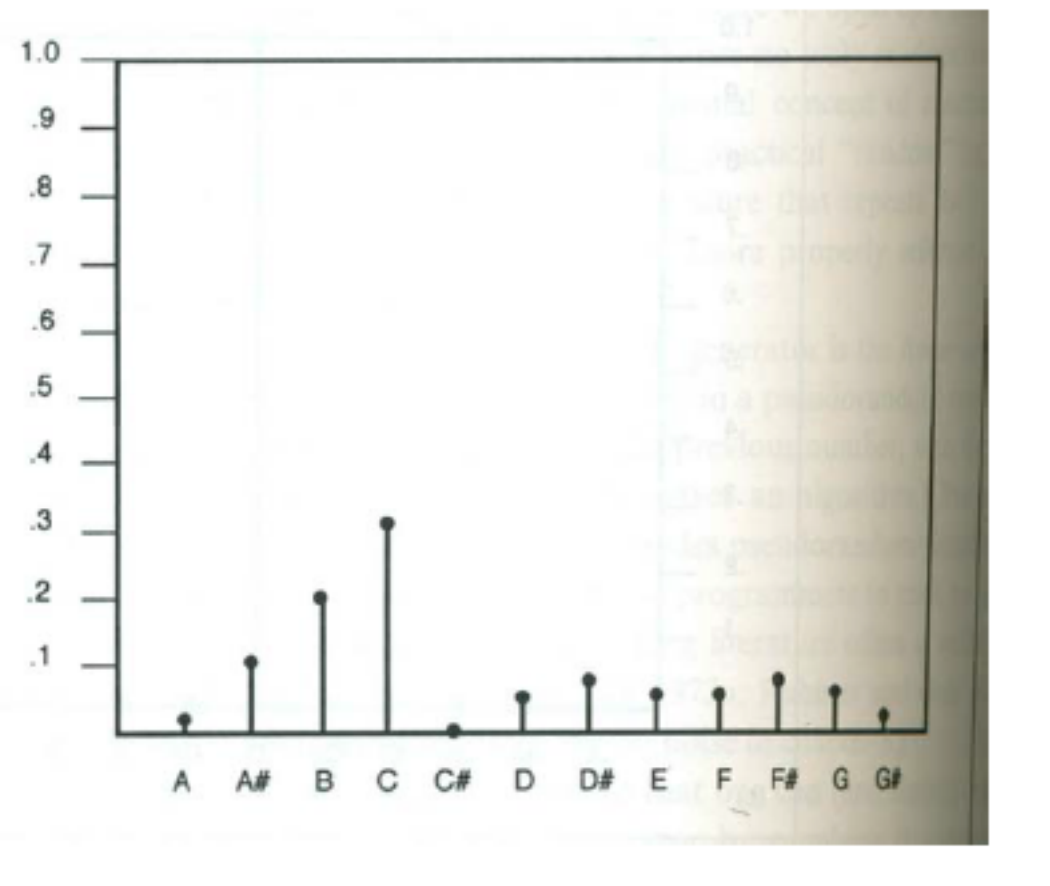
  - **x/(width/2) -> cutoff multiplier**
  - **map((y+x),0,height+width,60,200) –> fundamental frequency**

RandomWalkLevy

Multi voice Levy Flight

FURTHER CONSIDERATIONS

- Probability distribution can be learnt from a corpus of data



- E.g. Analysing the frequencies of notes in a set of composition we can retrieve the probability for each note
- The generated walk can follow the learnt probability distribution

NOISE

- Specifically, a sequence of random values is called **noise**
- Probability distribution of data generation allows to define the correlation between consequent values in the noise
- Uniform distribution -> White noise -> no correlation between consequent values

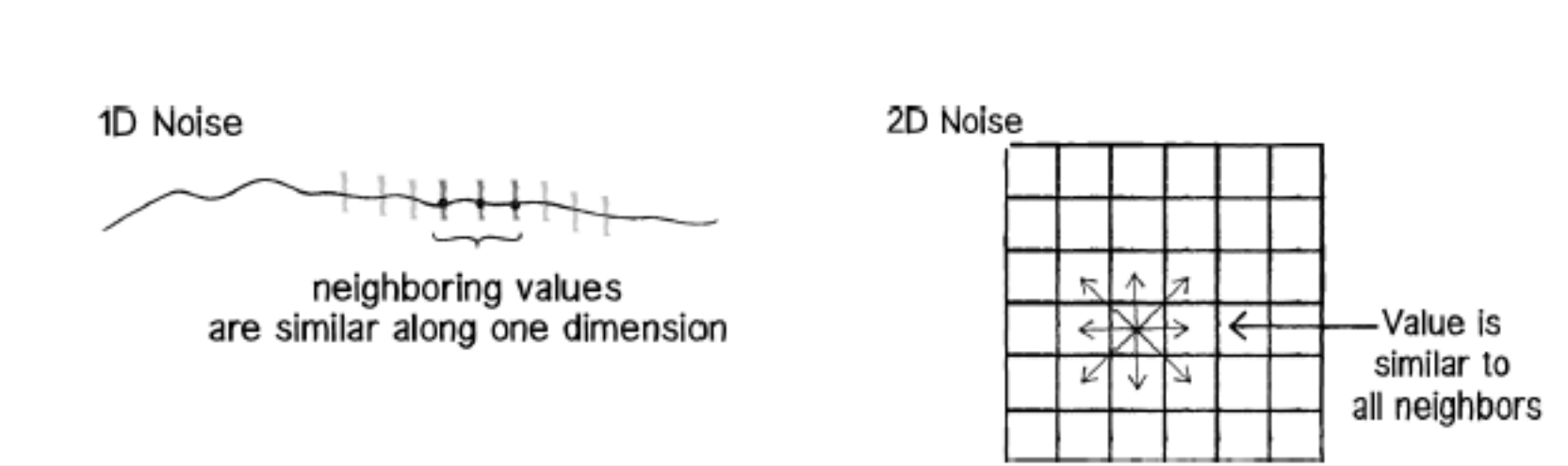
In SuperCollider: **Pwhite, Pbrown**

NOISE AND COMPUTER GRAPHICS

- Noise is a fundamental elements in creative computing, however the standard definition is not suitable to model natural events
- It seem that in nature very few phenomena can be really described by uniform probability distribution
- In the early 1980s, Ken Perlin wan working on the original ***Tron*** movie
- His task: build mathematical models to to generate various effects with natural qualities, such as clouds, landscapes, and patterned textures like marble
- Those qualities exhibits patterns but at the same time a certain degree of randomness

NOISE AND COMPUTER GRAPHICS

- The **perlin noise** tends to keep a certain degree of similarity between sequent values



- In Processing -> **noise()**
- Perlin noise is defined in an infinite n-dimensional space, in which each pair of coordinates corresponds to a fixed semi-random value (fixed only for the lifespan of the program)

PerlinNoise2D

PerlinNoiseWalkAcceleration

GRAMMAR-BASED AGENTS

GRAMMARS

- Let's think about natural language
- Human languages has three dimension: **syntax**, semantics and pragmatics
- Syntax
  - The language is composed by language **elements** (verbs, adjectives, etc.)
  - Elements are combined in sentences following **rules**
  - The set of elements and the set of rules compose a **grammar**
  - The **syntax** of a Language can be formalized using a formal grammar

- With a grammar:
  - Possible to assert if a sentence is part of the language or not
  - Generate new sentences following the rules

GRAMMARS

- A formal grammar is a set of rules to expand high-level symbols (**non-terminal symbols** or **sentences**) into more detailed sequences of symbols (**terminal symbols** or **words**) representing elements of formal languages.
- Words are generated by repeatedly applying rewriting rules, in a sequence of so-called **derivation steps**.
- Grammars are suited to represent systems with hierarchical structure, which is reflected in the recursive application of the rules
- Rules are:

LEFT SIDE -> RIGHT SIDE

- The left side gets replaced by the right side
- The left and right side can be either non-terminal or terminal symbols

GRAMMARS - EXAMPLE

A set of non-terminal symbols, V

- S :Sentence
- VP :Verbal Phrase
- NP :Noun Phrase
- V :Verb
- PP :Prepositional Phrase
- AP :Adjective Phrase
- Adv :Adverb
- P :Preposition
- Det :Determinant
- N :Noun

A set of rules, R

- S → NP VP
- VP → V (NP) (PP)
- AP → (Adv) A (PP)
- PP → P NP
- NP → (Det) (AP) N (PP)

() = optional

S = initial symbols

GRAMMARS - EXAMPLE

A terminal symbols, V_T

- V_T = {“Dolly”, “Bob”, “cat”, “dog”, “house”, “a”, “the”, “follows”, “likes”, “knows”, “small”, “big”, “nice”, “hungry”, “very”, “extremely”, “very”, “extremely”, “in”, “for”, “to”}

Non-terminal to terminal rewrite rules

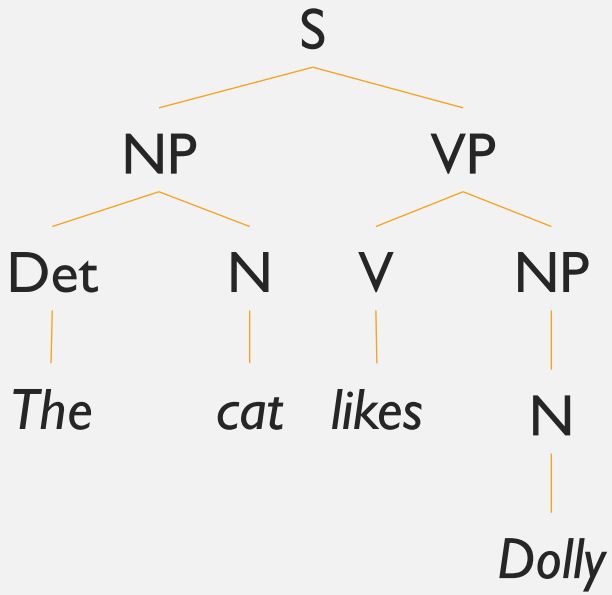
- N → “Dolly”, “Bob”, “cat”, “dog”, ”house”
- Det → “a”, ”the”
- V → “follows”, “likes”, “knows”
- A → “small”, ”big”, ”nice”, “hungry”
- Adv → “very”, “extremely”
- P → “in”, ”for”, ”to”

GRAMMARS - DEFINITION

- A grammar is defined as G=[V,V_T, R, S]
- V = Set of non terminal symbols
- V_T= Set of terminal symbols
- R = Set of rules
- S = Initial symbol

Check if a sentence belong to the language

The cat likes Dolly



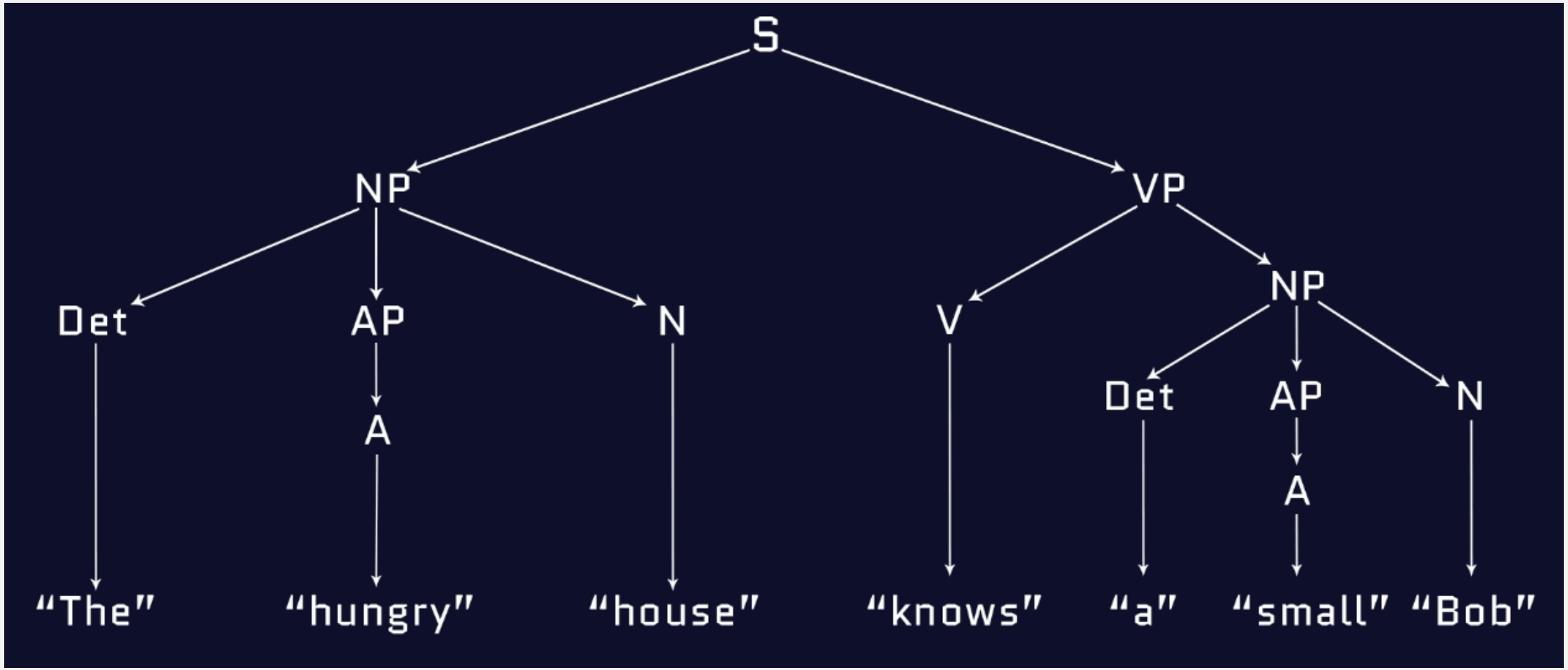
GRAMMARS - DEFINITION

Generate new sentences

S	:initial symbol
NP VP	:applying rule S → NP VP
Det N VP	:applying rule
“the” N VP	:applying Det → “the”
“the” “dog” VP	:applying N → “dog”
“the” “dog” V NP PP	:applying VP → V NP PP
“the” “dog” “follows” NP PP	:applying V → “follows”
“the” “dog” “follows” Det AP N PP	:applying NP → Det AP N
“the” “dog” “follows” “a” AP N PP	:applying Det → “a”
“the” “dog” “follows” “a” Adv A N PP	:applying AP → Adv A
“the” “dog” “follows” “a” “very” A N PP	:applying A → “big”
“the” “dog” “follows” “a” “very” “big” N PP	:applying A → “big”
“the” “dog” “follows” “a” “very” “big” “cat” PP	:applying N → “cat”
“the” “dog” “follows” “a” “very” “big” “cat” P NP	:applying PP → N P
“the” “dog” “follows” “a” “very” “big” “cat” “in” NP	:applying P → “in”
“the” “dog” “follows” “a” “very” “big” “cat” “in” Det N	:applying NP → Det N
“the” “dog” “follows” “a” “very” “big” “cat” “in” “the” N	:applying Det → “the”
“the” “dog” “follows” “a” “very” “big” “cat” “in” “the” “house”	:applying N → “house”

GRAMMARS - DEFINITION

Syntax correct but not the semantics



But … in some context not-semantically correct sentences are creative sentences

GRAMMARS - DEFINITION

- In the case more rules shares the left-side it is possible to define the probabilities to each rule to be applied

R_1: S → NP VP NP	$p(R_1)=0.1$, rare
R_2: S → NP VP	$p(R_2)=0.3$, occasional
R_3: S → NP	$p(R_3)=0.6$, common

- The Grammar and the probability can be learnt by a corpus of data
- Many algorithms has been proposed

GRAMMARS – MUSIC COMPOSITION

- Grammars are suited to represent systems with hierarchical structure, which is reflected in the recursive application of the rules

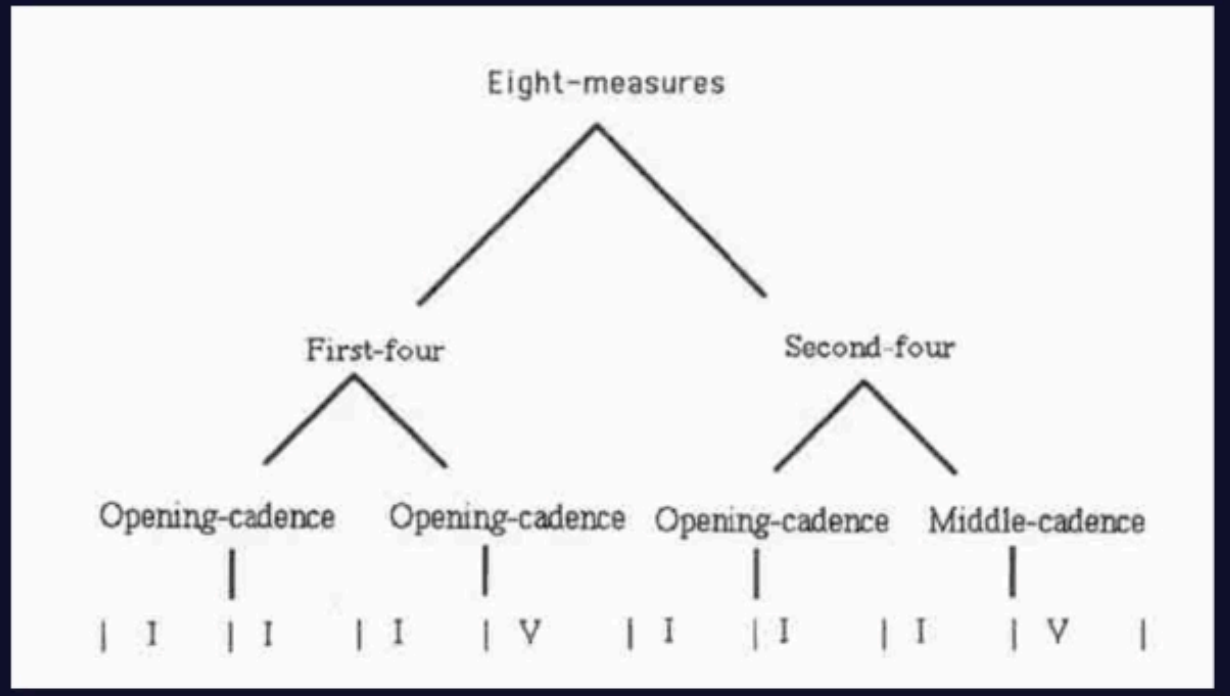
- As hierarchical structures can be recognized in most styles of music, it is hardly surprising that formal grammar theory has been applied to analyse and compose music for a long required for music composition

- First attempts to use grammas for music and relies on hand-made definition of symbols and rules

GRAMMARS – MUSIC COMPOSITION

A *Context-Free* Grammar for Generating 8 Measure Tonal Chord Sequences

Eight measures	→	First four	Second four
First four	→	Opening cadence	Opening cadence
	→	Opening cadence'	Opening cadence
Second four	→	Opening cadence	Middle cadence
		Opening cadence'	Closing cadence
		Middle cadence	Middle cadence
		Middle cadence	Closing cadence
Opening cadence	→	| I	| I |
	→	| I	| V |
Opening cadence'	→	| I	| III |
	→	| I	| IV |
Middle cadence	→	| I	| V |
	→	| V	| V |
Closing cadence	→	| I	| I |
		| IV	| I |
		| V	| I |



IMPRO-VISOR

- In 2007 Keller and Morrison created the Impro-visor
- Educational grammar-based software for automatic Jazz solo melody creation
- *"Impro-Visor was designed to assist the soloist, who may not yet have a strong theory background, in the process of creating solos"*

- The grammar was built manually based on jazz music theory
- Later on the introduced a tool for an evaluation from user to adjust probability distributions in rules selection

https://www.youtube.com/watch?v=1_AadO_A0BU

IMPRO-VISOR - RHYTHMIC SEQUENCES

- **1° Step – generate coherent skeletal rhythmic sequences**

- Terminal alphabet is **{ h, q },** representing half-notes and quarternotes
- Non-terminal alphabet is **{ S, M, H }**
  - S represents a sequence of one or more measures
  - M represents a measure
  - H represents a half-measure
- The last two roles allow the syncopation

- $S \rightarrow M$
- $S \rightarrow MS$
- $M \rightarrow HH$
- $H \rightarrow h$
- $H \rightarrow qq$
- $S \rightarrow HS$
- $S \rightarrow qS|$

IMPRO-VISOR - RHYTHMIC SEQUENCES

- The complete set of terminal symbols
- There is a complex sequence of rules to generate the terminal symbols

- **1** a whole-note
- **2** a half-note
- **4** a quarter-note
- **4.** a dotted quarter-note
- **8** an eighth-note
- **16** a sixteenth-note
- **4/3** a quarter-note triplet
- **8/3** an eighth-note triplet
- **16/3** a sixteenth-note triplet

IMPRO-VISOR - RHYTHMIC SEQUENCES

- An example of a rhythmic sequence

- $S \rightarrow M \rightarrow HH \rightarrow hH \rightarrow hh$
- $S \rightarrow M \rightarrow HH \rightarrow hH \rightarrow hqq$
- $S \rightarrow M \rightarrow HH \rightarrow qqH \rightarrow qqh$
- $S \rightarrow MS \rightarrow HHS \rightarrow qqHS \rightarrow qqhS \rightarrow qqhHH \rightarrow qqhqqH \rightarrow qqhqqqq$

IMPRO-VISOR - MELODY SEQUENCES

- **2° Step – once generated a rhythmic skeleton, we can provide a tone for each chord**
- Give a chord, notes can be:
- Terminal alphabet:

1) **C** a chord tone.
2) **L** a color tone.
3) **A** an approach tone.
4) **H** a "helpful" tone, defined to be any of one of the above.
5) **S** a scale tone, a member of a scale designated as being compatible with the chord.
6) **X** an arbitrary tone.
7) **R** a rest.

- **Chord tones**: tones of the current chord
- **Color tones**: tones that are not in the chord, but which are individually sonorous with it.: "tensions"
- **Approach tones**: non-chord tones that make good transitions to chord-tones or color tones
- **Other**: tones that do not fall into one of the categories above

A8/3 designates an approach tone with a duration of a eighth note triplet.

IMPRO-VISOR - MELODY SEQUENCES

1 is a duration string representing a whole-note
2 is a duration string representing a half-note
4 is a duration string representing a quarter-note
8 is a duration string representing an eighth-note
16 is a duration string representing a sixteenth-note
A8/3 designates an approach tone with a duration of a eighth note triplet.

1) **C** a chord tone.
2) **L** a color tone.
3) **A** an approach tone.
4) **H** a "helpful" tone, defined to be any of one of the above.
5) **S** a scale tone, a member of a scale designated as being compatible with the chord.
6) **X** an arbitrary tone.
7) **R** a rest.

IMPRO-VISOR - MELODY SEQUENCES

- The grammar for melody generation is like

Probability

$$
\begin{array}{llll}
P(0) \rightarrow empty & [\ 1\ ] & & \\
P(1) \rightarrow Q1 & [\ 1\ ] & & \\
P(2) \rightarrow Q2 & [\ 1\ ] & & \\
P(3) \rightarrow Q2 & Q1 & [\ 1\ ] & \\
P(n) \rightarrow Q2 & P(n-2) & [\ .25\ ] & \\
P(n) \rightarrow Q4 & P(n-4) & [\ .75\ ] &
\end{array}
$$

Constraint: Number of quarter notes

IMPRO-VISOR - MELODY SEQUENCES

- The remaining set of rules

$$
\begin{aligned}
Q4 &\rightarrow Q2 \quad V4 \quad V4 \quad [\ 0.52\ ] \\
Q4 &\rightarrow V8 \quad N4 \quad N4 \quad N4 \quad V8 \quad [\ 0.01\ ] \\
Q4 &\rightarrow V4 \quad Q2 \quad V4 \quad [\ 0.47\ ] \\
Q2 &\rightarrow N2 \quad [\ 0.06\ ] \\
Q2 &\rightarrow V4 \quad V4 \quad [\ 0.6\ ] \\
Q2 &\rightarrow V8 \quad N4 \quad V8 \quad [\ 0.12\ ] \\
Q2 &\rightarrow H4. \quad N8 \quad [\ 0.16\ ] \\
Q2 &\rightarrow H4/3 \quad H4/3 \quad H4/3 \quad [\ 0.06\ ] \\
Q1 &\rightarrow C4 \quad [\ 1\ ] \\
V4 &\rightarrow N4 \quad [\ 0.22\ ] \\
V4 &\rightarrow V8 \quad V8 \quad [\ 0.72\ ] \\
V4 &\rightarrow H8/3 \quad H8/3 \quad H8/3 \quad [\ 0.05\ ] \\
V4 &\rightarrow H8/3 \quad H8/3 \quad A8/3 \quad [\ 0.01\ ] \\
V8 &\rightarrow N8 \quad [\ 0.99\ ] \\
V8 &\rightarrow H16 \quad A16 \quad [\ 0.01\ ]
\end{aligned}
$$

$$
\begin{aligned}
N2 &\rightarrow C2 \quad [\ 1\ ] \\
N4 &\rightarrow C4 \quad [\ 0.5\ ] \\
N4 &\rightarrow L4 \quad [\ 0.2\ ] \\
N4 &\rightarrow S4 \quad [\ 0.5\ ] \\
N4 &\rightarrow A4 \quad [\ 0.01\ ] \\
N4 &\rightarrow R4 \quad [\ 0.25\ ] \\
N8 &\rightarrow C8 \quad [\ 0.4\ ] \\
N8 &\rightarrow L8 \quad [\ 0.2\ ] \\
N8 &\rightarrow S8 \quad [\ 0.4\ ] \\
N8 &\rightarrow A8 \quad [\ 0.01\ ] \\
N8 &\rightarrow R8 \quad [\ 0.1\ ]
\end{aligned}
$$

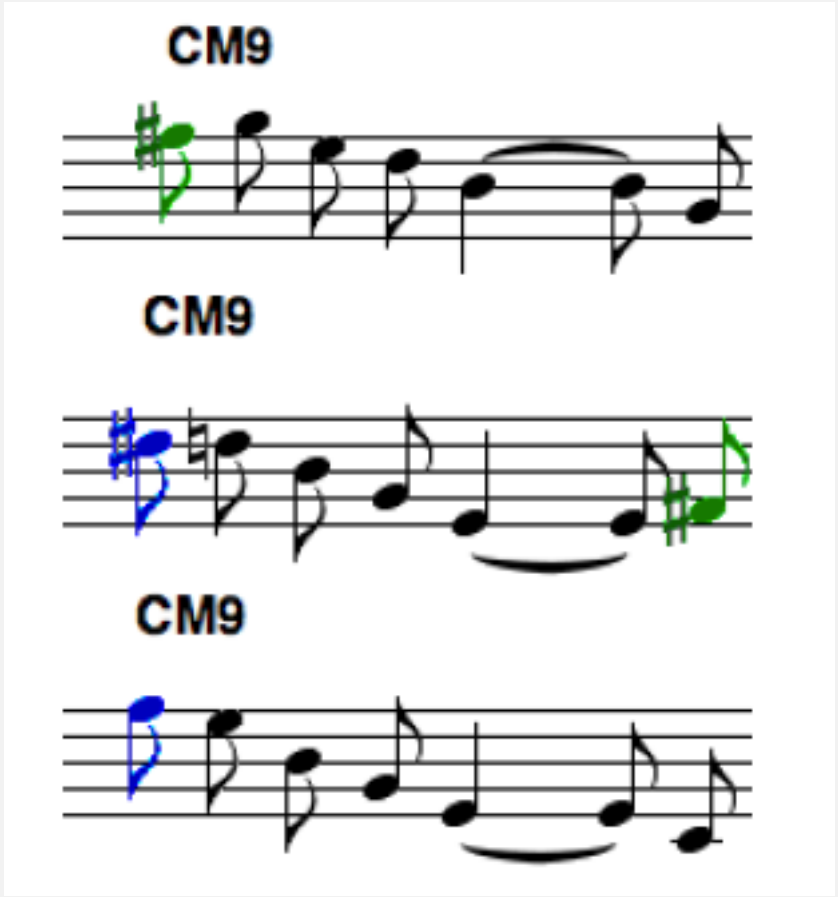
IMPRO-VISOR - MELODY SEQUENCES

- A possible sequence

$$A8 \quad L8 \quad S8 \quad C8 \quad H4. \quad S8$$

$$approach, color, scale, helpful, scale$$

- Give a C major 9



GRAMMAR-BASED COMPOSITION - EXAMPLE

To make the composition more intriguing, we need to have more variety in words and rules

- **Terminal symbols (words)**
  - **w:** whole note
  - **h**: half-note, i.e., half measure
  - **q**: quarter-note
  - **o**: quaver/ octave-note
  - **${w, h,q,o}**: corresponding pause
  - **t{h,q,o}** corresponding note in a triplet
- **Non-Terminal symbols (sentences)**
  - **S** → M, SM
  - **M** → HH, QHQ, w, $w, th th th
  - **H** → QQ, OQO, h, $h, tq tq tq
  - **Q** → q, oo, to to to, $q
  - **O** → o, $o
- **5 instruments**

Multitrack.mp3

L-SYSTEMS

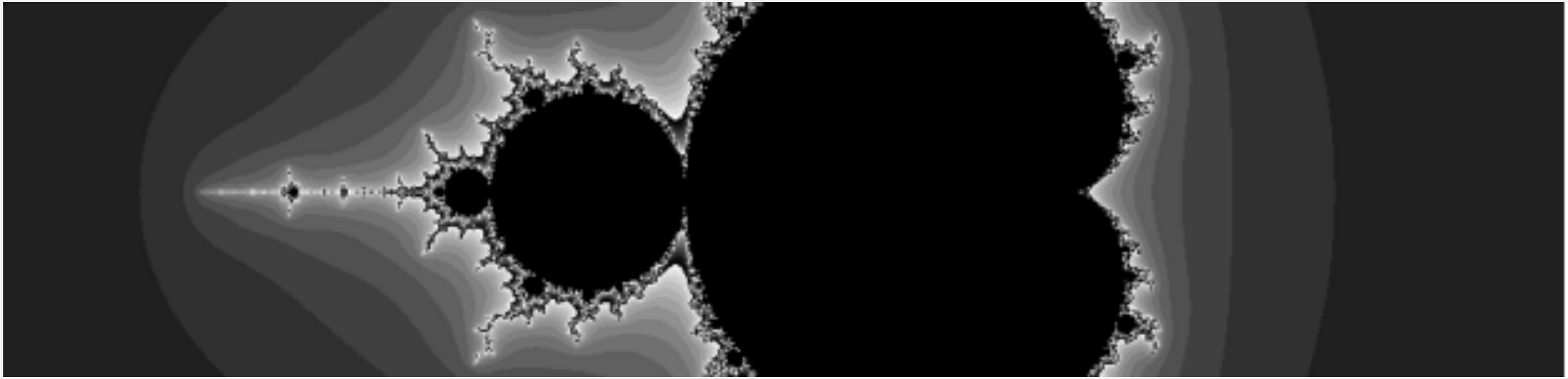
L-SYSTEMS

- In 1968, Hungarian botanist Aristid Lindenmayer developed a grammar-based system to model the growth patterns of plants
- In L-system there is no difference between terminal symbols and non-terminal symbols
- All the symbols corresponds to an action

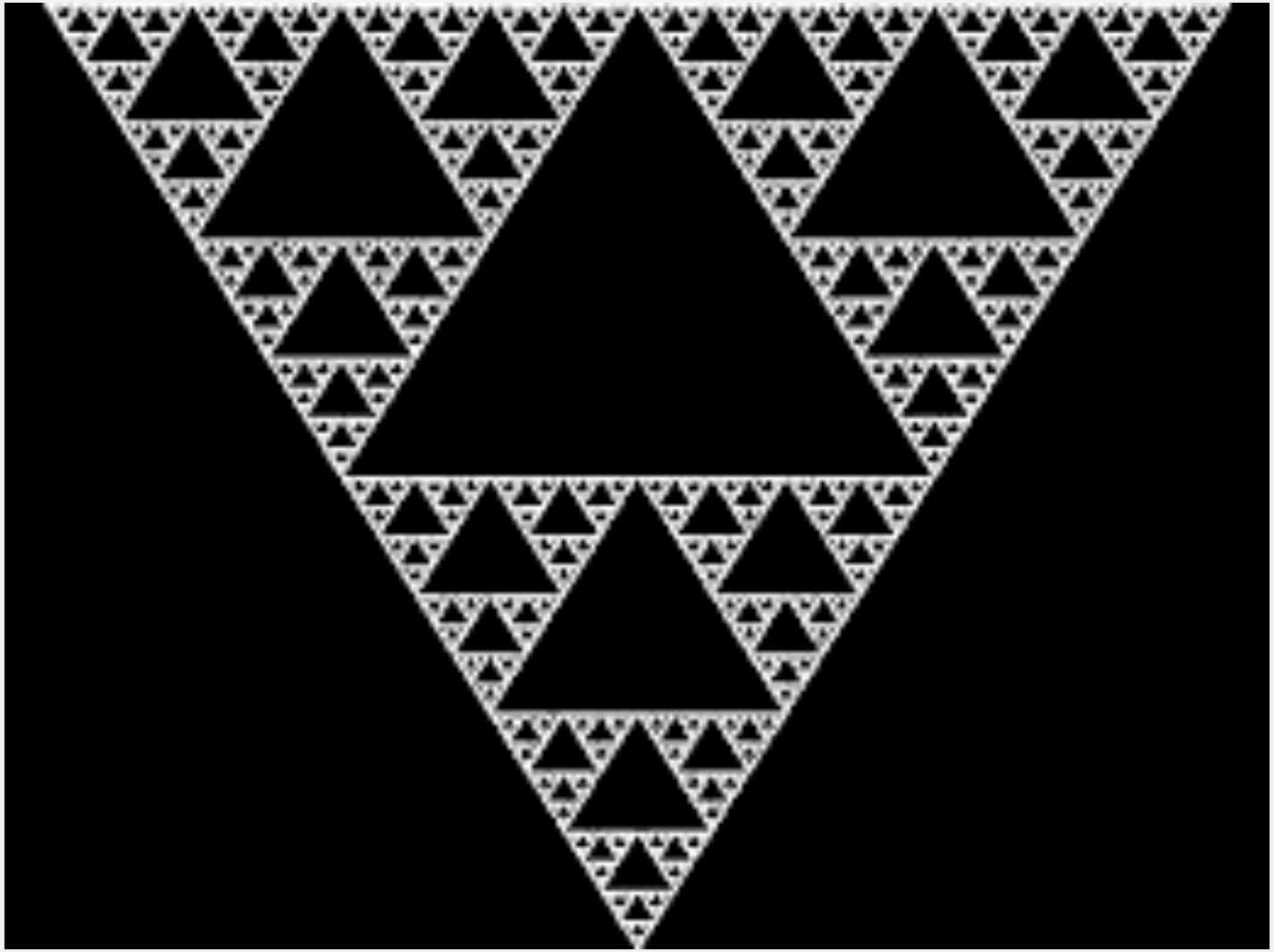
- L-System is defined as: **L=(v, w, P)**
- v = an alphabet of a finite set of symbols
  - v^* = set of all possible sequences generated from v (Kleene Closure)
  - $v^+ = v^* /\{\} = v^*$without the empty sequence
- w = seed/initiator $\in v^+$
- P = finite set of rules Z-> X where $Z \in v$, and the successor $X \in v^*$

L-SYSTEMS

- L-System are very power in generating fractals
- The fractals is defined "**a rough or fragmented geometric shape that can be split into parts, each of which is (at least approximately) a reduced-size copy of the whole**."



L-SYSTEMS



L-SYSTEMS

- A classical L-System alphabet

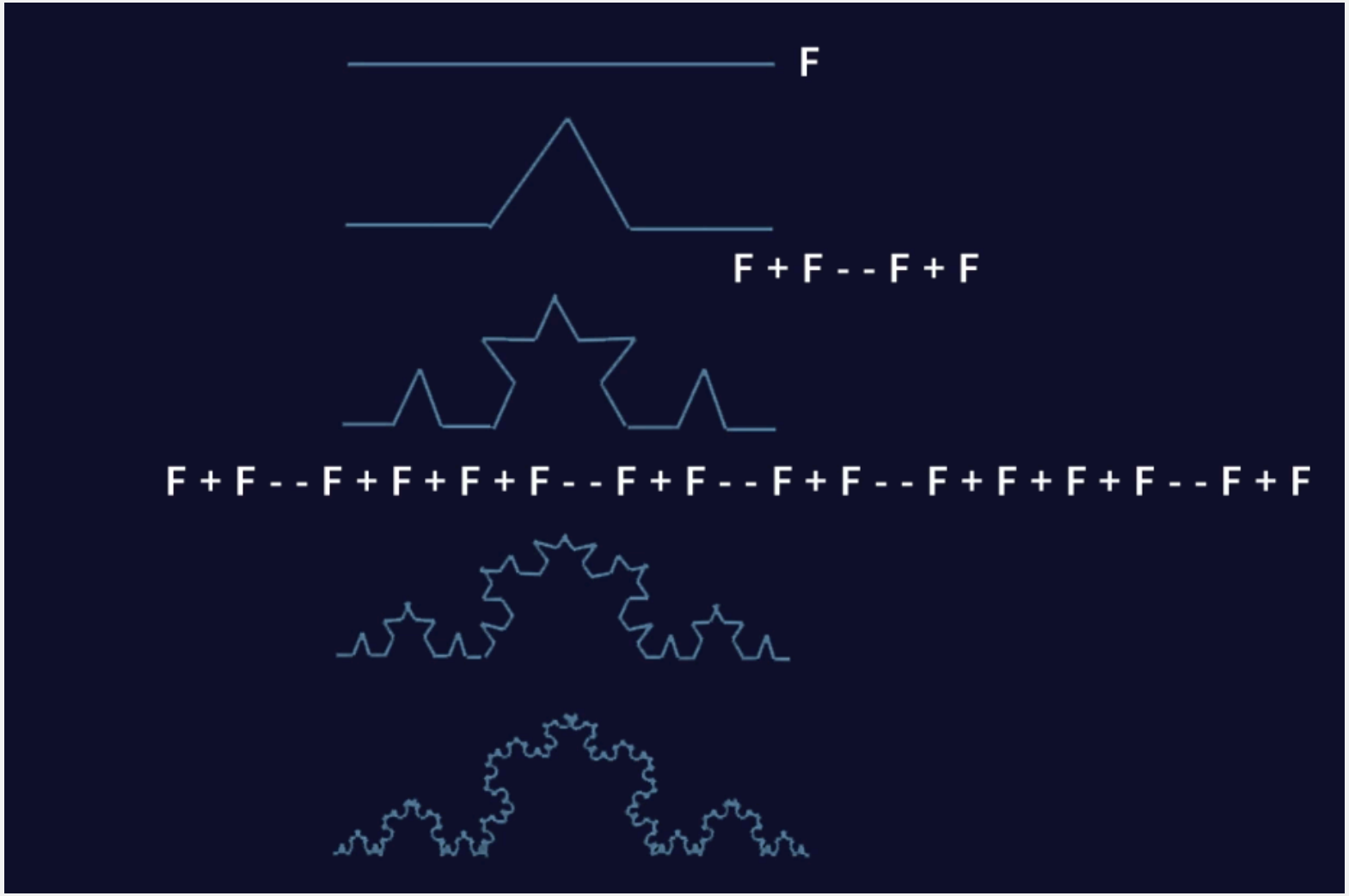
- F: Draw a line and move forward
- G: Move forward (without drawing a line)
- +: Turn left
- -: Turn right
- [: Save current location
-]: Restore previous location

- There is the need to define to how the actions corresponds in terms of drawing. Example:
  - Turn left and turn right: 60°
  - Move: 10 pixels

THE KOCH CURVE

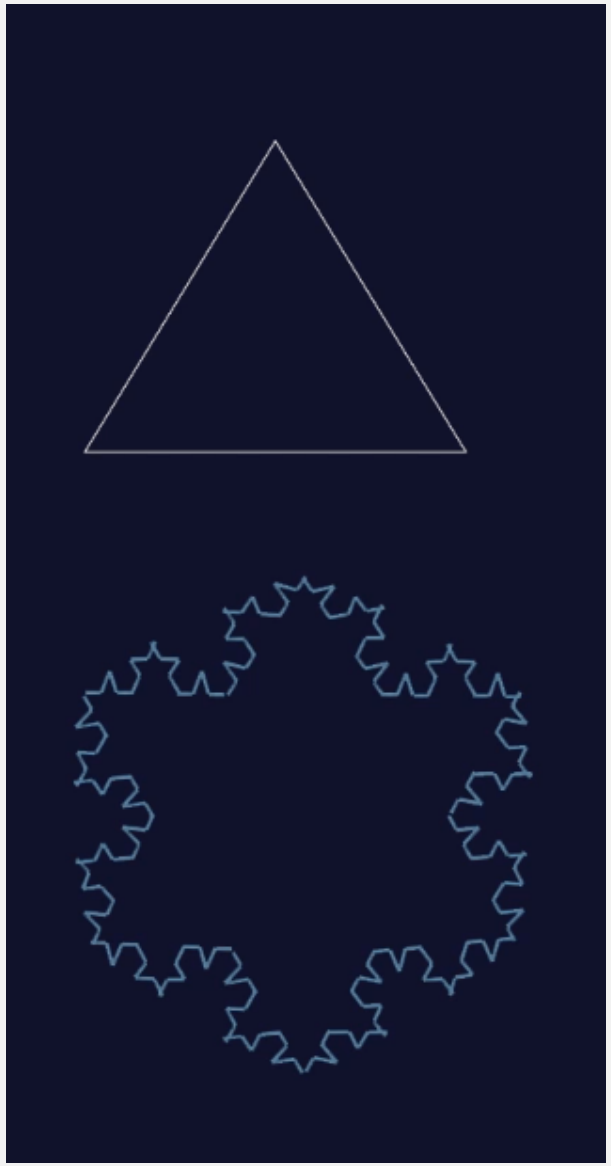
- v* = {F, +,-, {}}
- Rule: F –> F+F- -F+F
- Seed w = F

- F: Draw a line and move forward (length 2)
- +: Turn left (60°)
- -: Turn right (60°)



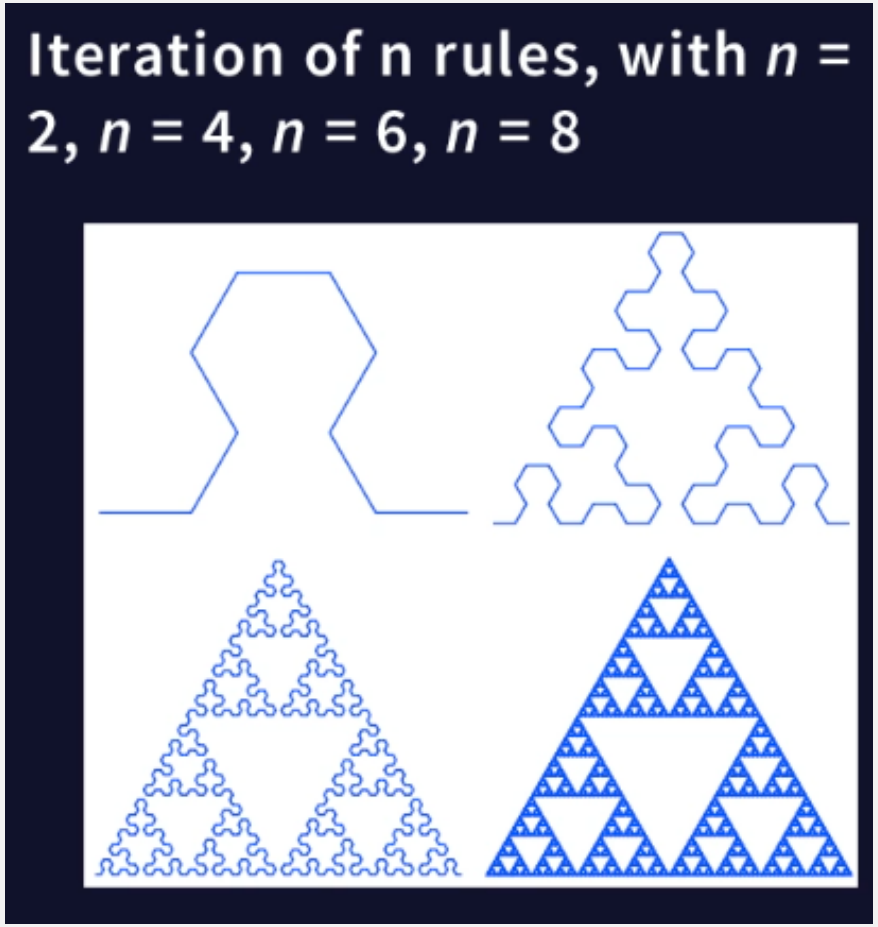
THE KOCH CURVE

- It is possible to take a complex shape a Seed
- Rule: F –> F+F- -F+F
- Seed = F - - F - - F
- Phi = 60° (angle)
- L = 2 (length)



SIERPINSKI TRIANGLE

- Vocabulary: F1, F2, +, -
- F1, F2: draw forward by L
- Rules:
  - F1 –> +F2-F1-F2+
  - F2 -> -F1+F2+F1-
- Seed = F
- Phi = 60° (angle)
- L = 2 (length)



- For n=2
- F1 -> +F2-F1-F2+ -> +-F1+F2+F1- - +F2-F1-F2+ - -F1+F2+F1-

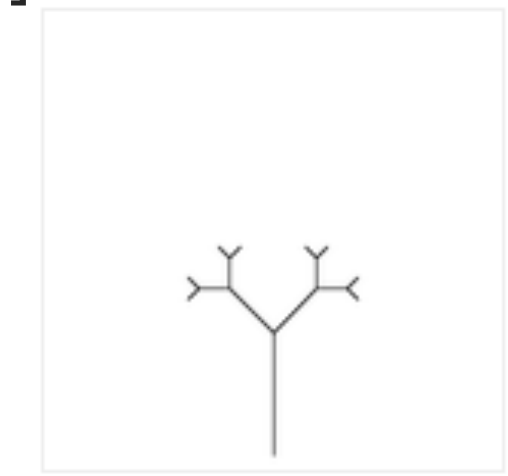
L-SYSTEMS – BINARY TREE

- F1: draw a line segment ending in a leaf
- F2: draw a line segment
- Rules:
  - F2 –> F2 F2
  - F1 –> F2 +[F1]-F1
- Seed = F1
- Phi = 45° (angle)
- L = 2 (length)

F1

F2+[F1]-F1

F2F2[F2+[F1]-F1]F2+[F1]-F1



L-SYSTEMS – FRACTAL PLANT

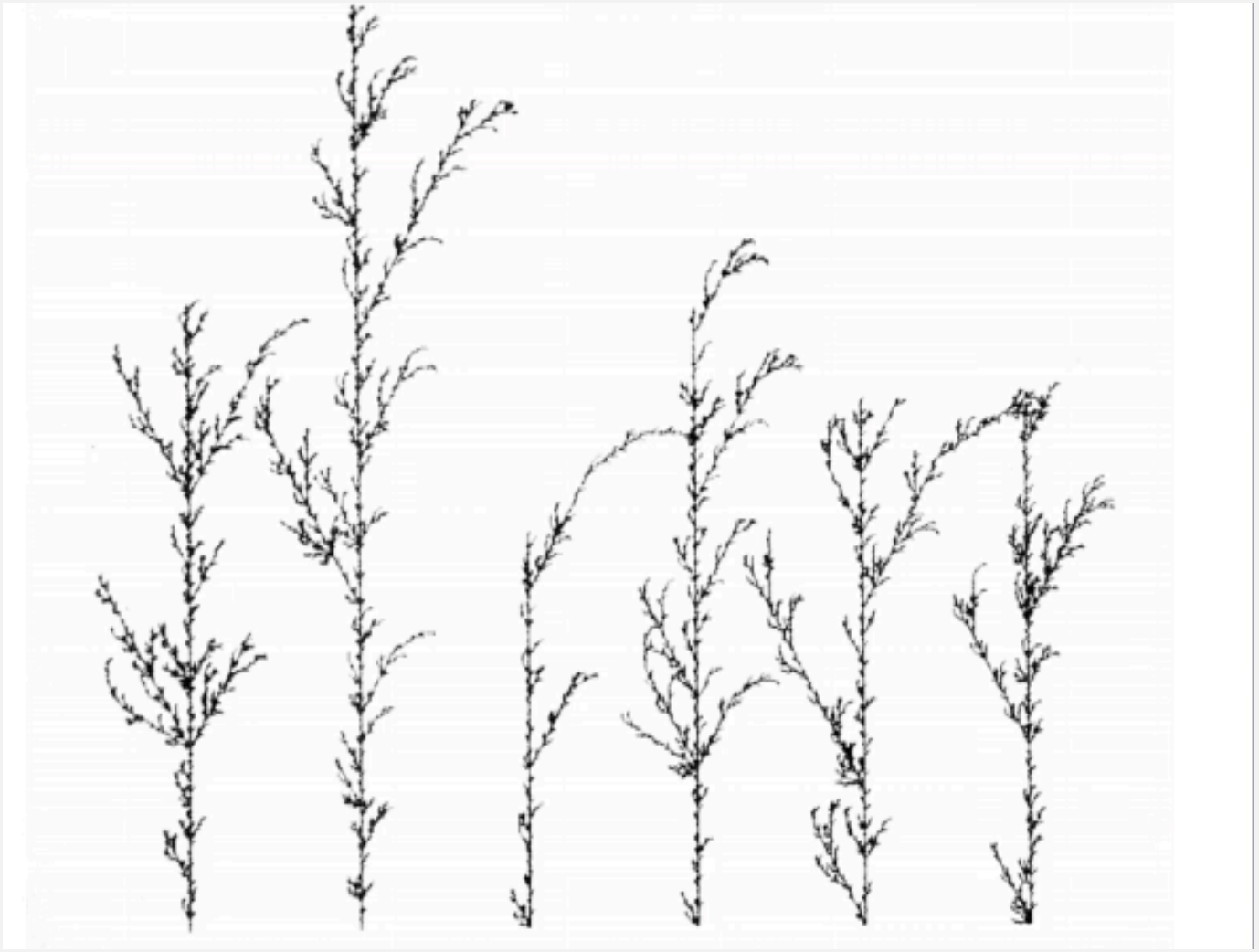
- Alphabet: X, F
  - X is not an action
- Rules:
  - (X → F+[[X]-X]-F[-FX]+X)
  - (F → FF)
- Seed = X
- Phi = 25° (angle)
- L = 2 (length)

LSystem_Tree

STOCHASTIC L-SYSTEMS

- In deterministic L-Systems there is one rule per symbol
- In stochastic L-Systems there can be more production for each symbols
- In this case a likelihood is associated to each rule

P1: F → F[[+F]]F[[-F]]F	p(P1)=0.33
P2: F → F[[+F]]F	p(P2)=0.33
P3: F → F[[-F]]	p(P3)=0.34



L-SYSTEM - EXAMPLE

Jon McCormack - Bloom

MATERIALS

- **References**
  - Robert M. Keller, David R. Morrison - A grammar approach to improvisation, Proceedings SMC'07, 4th Sound and Music Computing Conference, 11-13 July 2007, Lefkada, Greece
  - Jose David Fernandez, Francisco Vico, AI Methods in Algorithmic Composition: A Comprehensive Survey, Journal of Artificial Intelligence Research 48 (2013)
  - Chien-Hung Liu and Chuan-Kang Ting, Computational Intelligence in Music Composition: A Survey, IEEE Transactions on Emerging Topics in Computational Intelligence, December 2016,

- **Further readings**
  - Terzidis - Algorithms for Visual Design Using the Processing Language, Wiley Publishing, 2009 – Chapter 7
  - Daniel Shiffman – Nature of code, 2012 – Chapter 8
  - Curtis Roads – The Computer music Tutorial, 1996 – Chapter 19