POLITECNICO DI TORINO

Corso di Laurea Magistrale in Ingegneria Matematica

Tesi di Laurea Magistrale

Alberi frattali e applicazioni grafiche

Relatore
prof. Valeria Chiado' Piat

Candidato
Irene Leone

Anno Accademico 2020 - 2021

Alla mia famiglia.

Ringraziamenti

Questo traguardo rappresenta la fine di un percorso che mi ha regalato tante soddisfazioni ed emozioni belle.
Se mi guardo indietro vedo tante persone amiche che mi sono state vicino e che ormai sono parte fondamantale della mia vita.
Vedo Corso Lione 24, la cucina del Primo Piano, la mia camera piccola e perennemente disordinata. Vedo le nottate in aula studio, le feste a Natale a fine sessione. Vedo persone sdraiate in piazza Castello, vedo volti sbucare dalle camere ogni volta che passo. Vedo il sorriso di Emilio e Rossella ad accogliermi, l'ansia prima degli esami, la felicita' di un pasto condiviso.
Vedo una me insicura che piange su un aereo per Londra e poi ne prende un altro per Lussemburgo, questa volta cosapevole e col sorriso. Vedo una me felice camminare per le strade di Shoreditch con 2 euro sulla carta, mangiare pizza da un pound, passare le giornate nella library di Mile End. Vedo la nostalgia e la malinconia di percorsi che lasciano spazio ad altri. Vedo persone amiche che seguono i propri sogni. Pagine che inevitabilmente lasciano spazio ad altre, ma in ogni riga, in ogni parola, in ogni sillaba c'e' e ci sara' sempre l'impronta delle persone che mi hanno accompagnata in questi anni.
E credo profondamente che i sentimenti vadano oltre i confini temporali.
Quindi grazie amici, grazie Primo Piano perche' grazie a voi l'amicizia e' diventata la priorita' della mia vita. Grazie Mario, Marco, Giorgio, Antonella, Enrichino, Domenico, Davide e tutti voi. Vi sono profondamente grata.
Grazie Asier, perche' senza la tua fiducia e la tua mano tesa non ce l'avrei mai fatta.
Grazie Martina e Franci per essere diventate un punto fermo nella mia vita, per aver condiviso con me non solo la vostra casa, ma anche la vostra camera.
Grazie a tutti i miei amici dell'Erasmus e a quelli che hanno reso il mio tirocinio in Lussemburgo un bellissmo ricordo.
Grazie Zia Maria, Rita, Alessandra e Federica per essere state la mia famiglia.

E soprattutto grazie mamma, papa' e Ferdi per la fiducia, la liberta' di inseguire i miei sogni, la mano tesa quando ne avevo bisogno. Grazie a tutti i miei familiari, i Forliano e i Leone per essermi stati sempre vecini

Un rigraziamento speciale alla Professoressa Chiado', per il suo supporto, la sua disponibilita', la sua fiducia e il suo rispetto. E' raro incontrare professionisti cosi'!

Grazie a tutti e... ad maiora!

Indice

Introduzione

Questo elaborrato segna segna la fine di un percorso e rappresenta il valore che per me ha la matematica: la cultura del bello.
Questa tesi tratti di alberi frattali. Il capitolo uno da' una definizione tecnica dei frattali che trova la sua caratteristica maggiore nella proprieta' di autosimilarita'. Sono dapprima riportati dei concetti tecnici di teoria della misura, misura di Hasdirff e dimensione di Hausdorff.
Gli alberi sono l'esempio piu' lampante di frattali. Immaginamio un tronco di albero che si divide in due. Poi ancora dividiamo in due parti ciascun tronco figlio e cois' via. Ogni particolare, ogni ramo ha la stessa forma e proprieta' dell'albero stesso. Il dettaglio non e' altro che la riproduzione in scala dell'oggetto stesso.

Ma come definire matematicamente la riproduzione di albero? Attraverso la definizione di L-systems che non sono altro che un insieme di simboli e regole che definiscono la riproduzione di oggetti a partire da un assioma (ad esempio un tronco). La potenza degli L-system sta nel fatto che le regole di produzione possono essere definite in contemporanea, non un simbolo alla volta, ma n simboli contemporaneamente.
Gli alberi frattali non sono altro che L-system in cui associamo ad ogni regola una rapporesentazione grafica.
Tuttavia, gli L-system non includono la tri-dimensionalita', ne' tantomeno il concetto di vincoli spaziali. Negli alberi reali infatti, i rami non si riproducono tutti nello stesso modo e alo stesso tempo. Allora, nel capitolo 2.3 introduciamo il concetto di metric-driven L-systems, in cui la regola di produzione e' sempre associata a vincoli spaziali.
Per esprimere invece il concetto di tri-dimensionalita' applichaimo gli L-system ad ogni dimensione, e quindi nel capitolo 2.4 vengono descritti i system L-system estesi, in cui viene ripreso il concetto di edge, di cella e di wall che sono alla base delle riproduzioni cellulari.

Fin'ora abbiamo risposto alle seguenti domande:
1- Cosa? Alberi Frattali
2- Come? Secondo L-system
3- Dove e quando? Tramite i metric-driven Lystem e gli L-system estesi

Manca solo una domanda a cui vogliamo dare risposta in questo elaborato: perche'? Perche' proprio alberi farttali?
La risposta e' nel capitolo 2.5. Gli alberi frattali minimizzano l'energia intesa come resistenza del sistema. Partiamo con l'assunzione che uno schema di irrigazione abbia la forma di albero e quindi con proprieta' di autosimilirata', con flusso uguale per ogni sistema e grandezza del network invariante. Attraverso concetti di equivalenza algebrica andiamo a definire una misura sullo spazio dei flussi di irrigazione che ha proprieta' di continuita'. Introduciamo il concetto di equivalenza dei pattern di irrigazione. Un teorema fondamentale citato nella sezione 2.5.3 esprime la compattezza dei pattern di irrigazione, che esiste sempre un pattern di irragazione convergente ad un istogramma. Grazie all'equivalenza allora possiamo utilizzare una sequenza di istogrammi per minimizzare la funzione di costo. Abbaimo quindi le due proprieta' fondamentali per esistenza e unicita' del minimo che sono continuita' e compatezza. Allora esiste il minimo della funzione di costo, e' unico ed ha al forma di un albero, poiche' siamo partiti dall assunzione che i pattern di irrigazione hanno la forma di alberi frattali.

CAPITOLO 1

Geometria di oggetti frattali

1.1 Nozioni base di teoria della misura

Questo Capitolo ha come fonti [1], [2], [?] e [3] Sia X un qualunque insieme. Una collezione non vuota M di sottoinsiemi di X si dice σ-algebra di M se:

1. $\forall Q \in M \Longrightarrow X \setminus Q \in M$

2. $Q_1, Q_2, ... \in M \Longrightarrow \bigcup_{i=1}^{\infty} A_i \in M$

dunque se e' chiusa rispetto al complementare e all'unione numerabile. Ne segue che una σ-algebra e' chiusa anche rispetto all'intersezione numerabile.
Sia A una collezione di sottoinsiemi di X. A genera una σ-algebra $M(A)$ che e' l'intersezione di tutte le σ-algebra contenenti A ed e' dunque la piu' piccola σ-algebra contenente A.
Una misura μ e' una funzione definita su una σ-algebra $M(A)$ di sottoinsiemi A di X

$$\mu \colon \mathcal{M}(A) \to [0, +\infty]$$

che rispetta le seguenti proprieta':

$$\mu(\emptyset) = 0 \tag{1.1.1}$$

$$\mu(\bigcup_{i=1}^{n} A_i) = \sum_{i=1}^{n} \mu(A_i) \qquad \text{(proprieta' di additivita')} \tag{1.1.2}$$

$\forall A, A_i \in M$.
Dalla proprieta' di additivita' segue che μ e' una funzione crescente, ossia presi $A \subset A'$ allora $\mu(A) \leq \mu(A')$ dove $A, A' \in M$. Introducioamo ora il concetto di misura esterna ν che e' una misura indelibolita della proprieta' di addivitita'. Formalizzando, una misura esterna ν e' una funzione definita su X a valori in $[0, +\infty]$ per cui valgono le seguenti proprieta':

1.
$$\mu(\emptyset) = 0 \tag{1.1.3}$$

2.
$$\mu(A) \leq \mu(A') \qquad se \qquad A \subset A' \tag{1.1.4}$$

3.
$$\text{Proprieta' di subadditivita'} \qquad \mu(\bigcup_{i=1}^{n} A_i) \leq \sum_{i=1}^{n} \mu(A_i) \tag{1.1.5}$$

Un sottoinsieme C di X e' detto μ-misurabile (misurabile rispetto alla misura esterna μ) se
$$\mu(Z) = \mu(Z \cap C) + \mu(Z/E) \tag{1.1.6}$$
per ogni insieme $A \in X$.

Teorema
Se μ una misura esterna, l'insieme M degli insiemi μ-misurabili genera una σ-algebra e la restrizione di di μ a M e' una misura.
Dim

Sia (X, d) uno spazio metrico. Gli insiemi appartenenti alla σ- algebra generati dai chiusi di X sono detti insiemi boreliani di X. Per definizione di σ algebra, i boreliani includono gli aperti (essendo compelmentari dei chiusi), le unioni numerabili dei chiusi e le intersezioni numerabili degli aperti. Una funzione μ su X e' detta misura esterna metrica se $\mu(A \cap B) = \mu(A) + \mu(B)$ quando la distanza tra A e B e' positiva, ossia $\delta(E, F) = \inf\{d(x, y) : x \in A, y \in B\}$.
Teorema
Sia μ una misura esterna metrica su X. Allora tutti i sottoinsiemi boreliani di X sono μ-misurabili.
Dim

1.2 Misura di Hausdorff

Sia U un insieme non vuoto di dimensione n dello spazio Euclideo $\mathbb{R}^n$. Il diametro di U e' definito come la piu' grande delle distanze tra due punti $|U| = sup\{|x-y| : x, y \in U\}$.

Sia $\{U_i\}$ un insieme numerabile di diametro al piu' σ che ricopra un insieme F tale che $F \in \bigcup_{i=1}^{\infty} U_i$, dove $0 < |U_i| \leq \sigma$. Allora si dice che $\{U_i\}$ e' un σ-ricoprimento di F. Sia F un sottoinsieme di di $\mathbb{R}^n$ e sia s un numero non negativo. Definiamo per ogni $\delta > 0$

$$(F_\delta) = \inf\{\sum_{i=1}^{\infty} |U_i|^s : \{U_i\} \text{ è un } \delta\text{-ricoprimento di } F\} \tag{1.2.1}$$

Dunque al diminuire di σ l'insieme degli $\{U_i\}$ che soddisfa diminuisce. Allora, al tendere di δ a zero, (F_δ) aumenta.

Definiamo come hausd(F) il minite al tendere di δ a zero di (F_δ)

$$(F) = \lim_{\delta \to 0}(F_\delta) \tag{1.2.2}$$

Questo limite esiste ed e' la misura esterna s-dimensionale di Hausodrff di F. Se F_i e' un insieme numerabile di Boreliani, allora:

$$(\bigcup_{i=1}^{\infty} F_i) = \sum_{i=1}^{\infty}(F_i) \tag{1.2.3}$$

Ossia la misura di Hausdorff dell'unione di Boreliani e' la somma delle misure dei Boreliani.
La misura di Hausdorff riprende il concetto classico di area di una supercifie, intesa come la somma di quadrati di are 1 m^2.

Consideriamo ad esempio s=0 allora coincide con la misura che conta.

1.3 Dimensione di Hausdorff

Consideriamo la misura di Hausdorff (1.2) di un insieme $F \subset \mathbb{R}^n$

$$(F) = \inf\Big\{ \sum_{i=1}^{\infty} |U_i|^s : \ \{U_i\} \text{ sia un } \delta\text{-ricoprimento di } F\Big\}$$

Allora, per ogni $F \subset \mathbb{R}^n$ con $\delta < 1$, (F) e' una funzione non crescente all'aumentare di s.
Sia a ed s maggiori strettamente di zero e tali che $a > s$. Sia $\{U_i\}$ un δ-ricoprimento di F con $\delta < 1$ si ha che

$$\sum_{i=1}^{\infty} |U_i|^a = \sum_{i=1}^{\infty} |U_i|^{a-s} |U_i|^s \leq \delta^{a-s} \sum_{i=1}^{\infty} |U_i|^s \tag{1.3.1}$$

La precedente disuguaglainza rappresenta una contrazione dal momento che $\delta^{t-s} < 1$. Questo conferma che la funzione di Hausdorff e' non crescente.

$$\mathcal{H}^t_\delta(F) \leq \delta^{t-s}(F) \tag{1.3.2}$$

Se $\delta \to 0^+$ allora si ha che

$$\mathcal{H}^t = 0 \qquad \text{per } t > s \tag{1.3.3}$$

Disegnando il grafico di (F) al variare di s, si puo' notare l'esistenza di un punto in cui punto la funzione (F) passa da da ∞ a 0. Questo punto è la **dimensione di Hausdorff** di F $\dim_H F$.

Dunque:

$$(F) = \begin{cases} \infty & \text{se } 0 \leq s < \dim_H F \\ 0 & \text{se } s > \dim_H F \end{cases} \tag{1.3.4}$$

Se $s = \dim_H F$, (F) allora $0 \leq (F) \leq \infty$. Approfondiamo il concetto di dimensione di Hausdorff tramite il seguente esempio. Consideriamo F, disco di raggio unitario in $\mathbb{R}^3$.
Allora si ha

$$(F) = \infty$$

$$\mathcal{H}^2(F) = \frac{4}{\pi} area(F) = 4$$

$$\mathcal{H}^3 = \frac{6}{\pi} vol(F) = 0$$

Dunque la $\dim_H F = 2$ in quanto

$$(F) = \begin{cases} \infty & \text{se } s < 2 \\ 0 & \text{se } s > 2 \end{cases}$$

1.3.1 Proprieta' della dimensione di Hasdorff

1. *Monotonia.*
   Se $E \subseteq F$ allora $\dim_H E \leq \dim_H F$. Sia $s = F$ e $t = E$. Allora

$$t = E = \inf\{r \colon \mathcal{H}^r(E) = 0\}$$

Se $E \subset F$ allora

$$(E) = \begin{cases} 0 & \text{se } s \leq t \\ +\infty & \text{se } s > t \end{cases}$$

Sia per assurdo che $s < t$ e $(E) = +\infty$ per $s < t$ allora

$$(E) = \lim_{\delta \to 0^+} \inf\{\sum [U_i]^s\}$$

dove $\{U_i\}$ è un δ-ricoprimento di E, e, poiche' $E \subseteq F$ è parte di un δ-ricoprimento di F, se $(E) = \infty$ allora $(F) = \infty$, che è assurdo in quanto $F = s$.

2. *Stabilità numerabile.*
   Sia $F_1, F_2, \ldots$ una successione numerabile di insiemi allora

$$\bigcup_{i=1}^{\infty} F_i = \sup_{1 \leq i \leq \infty} \{F_i\} \tag{1.3.5}$$

   Poiche' la dimensione di Hasdorff gode della proprieta' di monotonia, allora ne segue che che $\bigcup_{i=1}^{\infty} F_i \geq F_j \ \forall j$.
   Segue che

$$(\cup F_i) = \inf\Big\{ \sum_i \sum_j \left|U_j^i\right|^s : \bigcup_j U_j^i \supset \cup F_i \Big\} \leq \sum_i \inf\Big\{ \sum_j |U_j|^s : \bigcup_j U_j \supset \cup F_i \Big\} = 0$$

   Allora $(\cup F_i) = 0$.

3. *Insiemi numerabili.*
   Sia F è un insieme numerabile. Allora $F = 0$. Sia F_i sia un punto, allora $\mathcal{H}^0(F_i) = 1$ e la $F_i = 0$. Pocihe' F è un insieme numerabile, e' assimilabile all'unione di infiniti punti, quindi $F = \bigcup_{i=1}^{\infty} F_i$. Dalla proprietà di stabilità numerabile si ha che $F = 0$.

4. *Insiemi aperti.*
   Sia $F \subset \mathbb{R}^n$ un aperto, allora $F = n$. Poiche' F è un aperto, allora contiene una palla n-dimensionale. Dalla monotonia segue che $F \geq n$ e che F è contenuto in un insieme numerabile di palle n-dimensionali. Dunque $F \leq n$. Dunque $F = n$.

1.4 Automisilarita'

In questa sezione si descrive la proprieta' fondamentale degli alberi frattali di autosimilarita'. Immaginiamo il tronco di un albero. Quest'ultimo si divide altri due rami assimilabili ad un tronco. A loro volta questi si sdoppiano in altri due,

facendo si che ogni singolo dettaglio sia, in scala, l'albero stesso. Questa proprieta' e' detta proprieta' di autosimilarita'.
In questa sezione vengono introdotte le principali nozioni di auto-similarità.

Sia $r = (r_1, r_2, ..., r_n)$ una successione finita di numeri positivi reali definti tali che generino una ratio list r in uno spazio metrico S. Una funzione e' detta di similarità con coefficiente k su uno spazio metrico (X,d), $f\colon X \to X$ se presi due punti $x, y \in X$ $d(x, y) = kd(f(x), f(y))$
Si definisce sistema di funzioni iterati, una successione di funzioni di similarita' $(f_1, f_2, ..., f_n)$ con $f_i\colon S \to S$ con coefficient r_i. Un insieme non nullo compatto $K \subset S$ è definito autosiile, ossia invariante/attrattore per $(f_1, ..., f_n) \iff K = f_1[K] \cup ... \cup f_n[K]$. Infine il valore di similarità s di una ratio list è un numero>0 tale che

$$r_1^s + .. + r_n^s = 1$$

e questo numero esiste ed e' unico, come dimostrato nel seguente:
Teorema
Consideriamo una ratio lost $(r_1, ..., r_n)$ con$r_i < 1$. Allora esiste ed e' unico s non negativo tale che

$$\sum_{i=1}^{n} r_i^s = 1 \tag{1.4.1}$$

Se $s = 0 \iff n = 1$ Sia $\Phi\colon [0, +\infty) \to [0, +\infty)$ tale che

$$\Phi(s) = \sum_{i=1}^{n} r_i^s = 1$$

Φ è una funzione continua, $\Phi(0) = n \geq 1$ e $\lim_{s\to\infty} \Phi(s) = 0 < 1$.
Per il teorema dei valori intermedi esiste almeno un punto s in cui la funzione sia 1. Dalla negativita' di Φ

$$\Phi'(s) = \sum_{i=1}^{n} r_i \log r_i$$

si evince che Φ è sempre una funzione decresente, condizione sufficiente per dimostrare l'unicita' di unn punto tale che $\Phi(s) = 1$. Allora preso $n > 1$, $\Phi(0) > 1$ $s \neq 0$.
La dimenzione di autosimilarita' di un insieme non nullo e compatto e' quel valore s tale che $K = \bigcup_{i=1}^{n} f_i[K]$ con $(f_1, ..., f_n)$ rappresenti il sistema di funzioni iterate aventi s come ratio list.

1.4.1 La metrica di Hausdorff

La metrica di Hausdorff e' utile per comprendere il legame tra la dimesione di Hausdorff e le altre dimensioni. Sia S uno spazio metrico, A e B contenuti S.
Viene detto che A e B hanno distanza di Hausdorff r se e solo se ogni punto di A

è distante r da almeno punto di B e se ogni punto di B è distante r da almeno un punto di A. La metrica di Hasdorff D esprime matematicamente questo concetto.

Se A è un insieme e sia $r > 0$. Definiamo intorno aperto di A quell' insieme

$$(A) = \{y\colon d(x,y) < r \text{ per almeno una } x \in A\} \tag{1.4.2}$$

la metrica di Hasdorff e' definita dalla seguente:

$$D(A,B) = \inf\{r > 0\colon A \subseteq (B)\ e\ B \subseteq (A)\} \tag{1.4.3}$$

Per convenzione $D(\emptyset) = \infty$. Consideriamo $A = 0$ e $B = [0\ \infty)$. Allora $D(A,B) = \infty$, che contraddice la definizione di metrica. Si dice in tal caso che D è una metrica estesa, dato che assume valori infiniti. Introducendo tuttavia dei vincoli, si ottiene una definizione di metrica rigorosa. Consideriamo dunque solo insiemi compatti e non nulli. Sia S uno spazio metrico. (S) e' definito come l'insieme di tutti i sottospazi non vuoti e compatti S.Sotto queste condizioni, si puo' dimostrare che D è una metrica su (S). Proprieta' che devono verificarsi perche' D sia una metrica su (S) :

1. $d(x,y) = 0 \iff x = y$
2. $d(x,y) = d(y,x)$
3. $d(x,y) \leq d(x,z) + d(z,y) \qquad \forall x,y,z \in (S)$

La proprietà 2) e' una conseguenza della definzione stessa della metrica.
Si dimostra ora che $D(A,B) = 0 \iff A = B$
($\Longleftarrow$) Se $A = B$ allora $\forall \epsilon > 0$ si ha $A \subseteq \mathcal{N}_\epsilon(B)$, dunque $D(A,B) = 0$.
($\Longrightarrow$) Siano $A, B \in (S)$ tali che $D(A,B) = 0$
Se $x \in A$, allora $\forall \varepsilon > 0$ si ha $x \in \mathcal{N}_\varepsilon(B)$, allora dist$(x,B)$= 0. Poiche' B è chiuso (essendo compatto), allora $x \in B$. Partendo da $x \in B$ si dimstra che $B \subseteq A$. Allora $A = B$
Proseguiamno ora con la dimostrazione della disuguaglianza triangolare. Siano $A, B, C \in (S)$ e sia $\varepsilon > 0$. Se $x \in A$, allora esiste $y \in B$ tale che $d(x,y) < D(A,B) + \varepsilon$ ed esiste $z \in C$ tale che $d(y,z) < D(B,C) + \varepsilon$. Allora $A \in$(C)$conr =$ D(A,B)+D(B,C)+ 2ε e C è contenuto in (A). $D(A,C) \leq D(A,B) + D(B,C) + 2\varepsilon \quad \forall \varepsilon > 0$. Segue che

$$D(A,C) \leq D(A,B) + D(B,C)$$

1.5 Funzioni di similarità e connessione con le dimensioni

Consideriamo f una trasformazione di similarità con fattore $\lambda > 0$. Se $F \subseteq$allora

$$(f(F)) = \lambda^s(F) \tag{1.5.1}$$

Dunque se F viene scalato, ossia dilatato o rimpicciolito, di un fattore λ, allora si avra' che anche la sua dimensione sara'scalata di λ^s. Consideriamo la misura di Hausdorff e la sua dimensione. Sia U_i un $\delta-$ricoprimento dell'insieme F. Applicando f alla definizione di metrica di hasdorff si che $f(U_i)$ è un $\lambda\delta$-ricoprimento si $f(F)$, dunque

$$\sum |f(F)|^s = \lambda^s \sum |U_i|^s$$

Allora

$$_{\lambda\delta}(f(F)) \leq \lambda^s(F)$$

Se si considera il limite dell'stremo inferiore della formula precedente si ha che

$$(f(F)) \leq \lambda^s(F)$$

.
La dimosrazione dell'implicazione inversa segue lo stesso ragionamento applicando pero' la funzione f^{-1}. Si otterra' in tal caso che

$$\lambda^s(F) \leq (f(F))$$

La tesi e' dunque dimostrata.
Si dice che che una funzione soddisfi la condizione di Hölder di esponente α se dati $c, \alpha > 0$

$$|f(x) - f(y)| \leq c\,|x - y|^\alpha \quad x, y \in F \tag{1.5.2}$$

Dunque:

$$\mathcal{H}^{\frac{s}{\alpha}}(f(F)) \leq c^{\frac{s}{\alpha}}(F) \tag{1.5.3}$$

Se $\alpha = 1$ 1.5.2 esprime la condizione di Lipschitzianeita'.
Allora per funzioni Lipshitziane vale la seguente:

$$(f(F)) \leq c^s(F) \tag{1.5.4}$$

Si ha dunque che λ è uguale alla costante di Lipschitz c.

Se f è un'isometria ossia tale che

$$|f(x) - f(y)| = |x - y|$$

ne segue che

$$(f(F)) \leq haus(F)$$

Dalla precedente ne segue che la misura di Hausdorff è invariante per rotazioni e traslazioni.
Si puo' dimostrare inoltre che la dimensione di Hausdorff è invariante per trasformazioni bi-Lipschitziane.

CAPITOLO 2

Alberi Frattali

2.1 Definizione di L-systems

Il seguente capitolo ha come fonti [4] , [5], [6] e [7]

Un L-system, anche detto Lindenmayer system, e' un tipo di sistema di riscrittura parallelo che e' in logica definito come la sostituzione di un oggetto con un altro seguendo un criterio formale.
Un L-system consiste in:
a) un insieme di simboli, a partire dai quale si generano delle stringhe
b) regole di produzione che regolano la trasformazione dei simboli in stringhe
c) assioma che e' l'elemento da cui ha inizio la costruzione

Gli L-system portano il nome del proprio inventore Aristid Lindenmayer che nel 1968 li uso' per decsrivere l'evoluzione di piante. Per la loro caratteristica di auto-similarita' vengono utilizzati per generare frattali.
Matematicamente gli L-systems vengono definiti con una terna di oggetti dalla funzione

$$G(V, \omega, P) \tag{2.1.1}$$

dove
- V (l'alfabeto) e' un insieme di simboli, alcuni dei quali non sono costanti cioe' non sono soggetti a trasformazioni, altri invece sono variabili e soggetti a sostituzione
- ω e' detto assioma ed e' un elemento di V che rappresenta lo stato iniziale - P e' un insieme di regole di produzione costituite da due stringhe (predecessore e successore). Per gli elementi che non compaiono come predecessori si sottintende la legge di identita' e vengono definiti costanti. Le regole di produzione sono applicate in maniera iterativa a partire dall'assioma. Una caratteristica degli L-system e' che molteplici regole vengono applicate simultaneamente. Per questa proprieta' vengono definiti "linguaggi formali" generati da una "grammatica for-

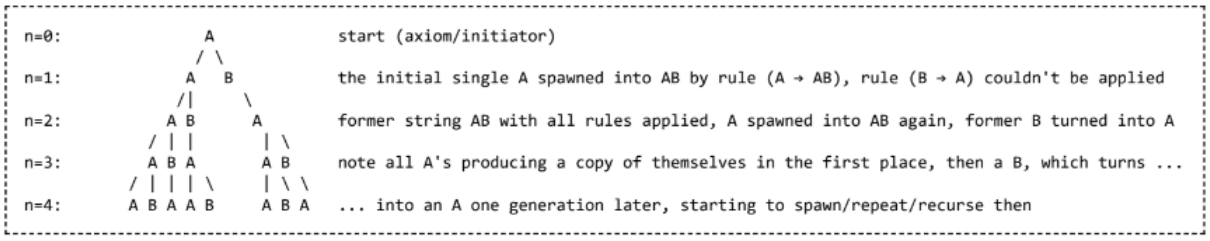

Figure 2.1: L-system

male". Se le regole si potessero applicare una alla volta allora sarebbero definiti linguaggi e non L-systems.

Esistono due tipi di L-system: i context-free, se ogni regola si riferisce al singolo simbolo e non al suo vicino, context-sensitive in caso contrario. Se esiste una e una sola regola per simbolo allora il sistema e' detto deterministico. Se invece ci sono piu' regole associate ad ogni simbolo, a ciascuna delle quali si associa una probabilita', allora il sistema e' detto stocastico.

2.2 Esempi di L-system

2.2.1 Esempio 1: Alghe

Il primo L-system di Lindenmayer e' stato utilizzato per modellare la crescita delle alghe, ed e' definito dai seguenti componenti:
1- Variabili: A,B
2- Costanti: nessuna
3- Assioma: A
4- Regole: (A $\rightarrow$ AB), (B $\rightarrow$ A)

Iterazioni:
n=0 : A
n=1 AB
n= 2 : ABA
n= 3 : ABAAB
n= 4 : ABAABABA
n= 5 : ABAABABAABAAB
n= 6 : ABAABABAABAABABAABABA
n= 7 : ABAABABAABAABABAABABAABAABABAABAAB

2.2.2 Esempio 2: Alberi frattali

Gli L-system utilizzati per descrivere alberi frattali presentano le seguenti caratteristiche:

1- Variabili: 0,1
2- Costanti: [,]
3- Assioma: 0
4- Regole: $(1 \rightarrow 11)$, $(0 \rightarrow 1\ [0]\ 0)$

A ogni iterazione ogni carattere della stringa subisce una trasformazione dettata dalle regole: ogni '1' si trasforma in '11', mentre ogni '0' si trasforma in '1 [0] 1'. Le parentesi diventano costanti. Il punto di partenza e' il carattere '0'.
Applicando le regole descritte a partire dall'assioma si ottiene:

n=0 0
n=1: 1[0]0
n=2: 11[1[0]0]1[0]0
n=3: 1111[11[1[0]0]1[0]0]11[1[0]0]1[0]0
...

Dall'esempio precedente possiamo vedere che la complessita' e la dimensione aumentano molto velocemente.
Associando ad ogni simbolo una operazione grafica si ottiene un immagine. Ad esempio:
0: disegna un segmento che termini in una foglia
1: disegna un segmento
: push della posizione e dell'angolo, ruota a sinistra di 45

: pop della posizione e dell'angolo, ruota a destra di 45

La differenza tra il push e il pop (a parte il verso della rotazione) e' che nel primo la posizione viene salvata in memoria, nell'ultimo no e si torna all'ultima posizione salvata. Dunque [1] si legge come: salva la posizione, ruota di 45, disegna un segmento, torna all'ultima poszione salvata.

2.2.3 Esempio 3: Piante frattali

Di seguito riportiamo un L-system utilizzato per disegnare una pianta.
1- Variabili: X, F
2- Costanti: + - []
3- Assioma: X

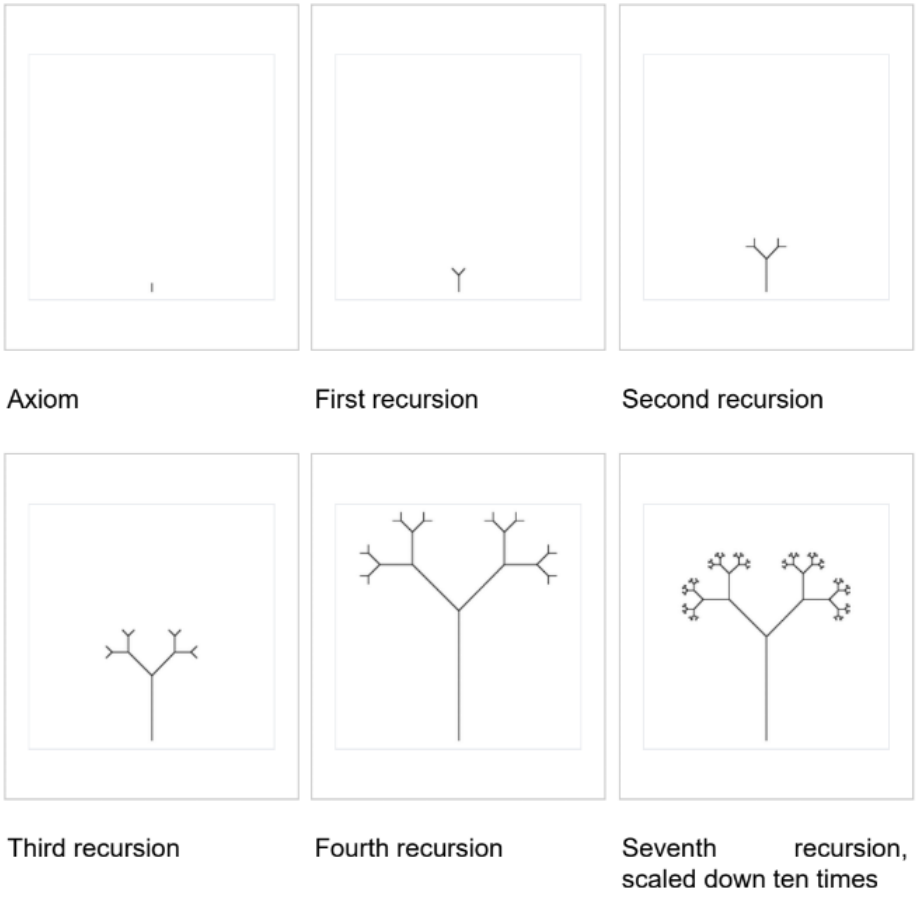

Figure 2.2: L system associati a rappresentazioni grafiche

Figure 2.3: L system generalizzato

4- Regole: (X → F+[[X]-X]-F[-FX]+X), (F → F)

F significa "disegna in avanti", - "gira a destra di 25 ", + "gira a sinistra di 25 ". X viene utilizzato per controllare lo stato del sistema e non appare graficamente. La parentesi quadra aperta salva la corrente posizione ed angolo, la chiusa torna all'ultima salvata.

2.3 Morfogenesi di strutture ad albero guidate da metriche e L-systems

La successione di strighe degli L-systsem viene utilizzata per rappresentare l'evoluzione di organismi. Tuttavia l'unica variabile indipendente che essi prendono in considerazione e' il tempo. Nell'evoluzione di organismi, ad esempio nella creascita di piante, c'e' un secondo fattore da prendere in cosiderazione che e' lo spazio, ossia la forma che non ha nessun impatto negli L-system sopra citati.
Per descrivere i processi morfogenici si utilizzano grammatiche "metric-driven". Queste ultime operano su celle complesse.Una metrica su celle complesse tiene in considerazione la distanza tra gli elementi, che cambia con la crescita e quindi con il tempo.

La figura 2.4 e' riportato un esempio di grammatica "metric driven". La re-

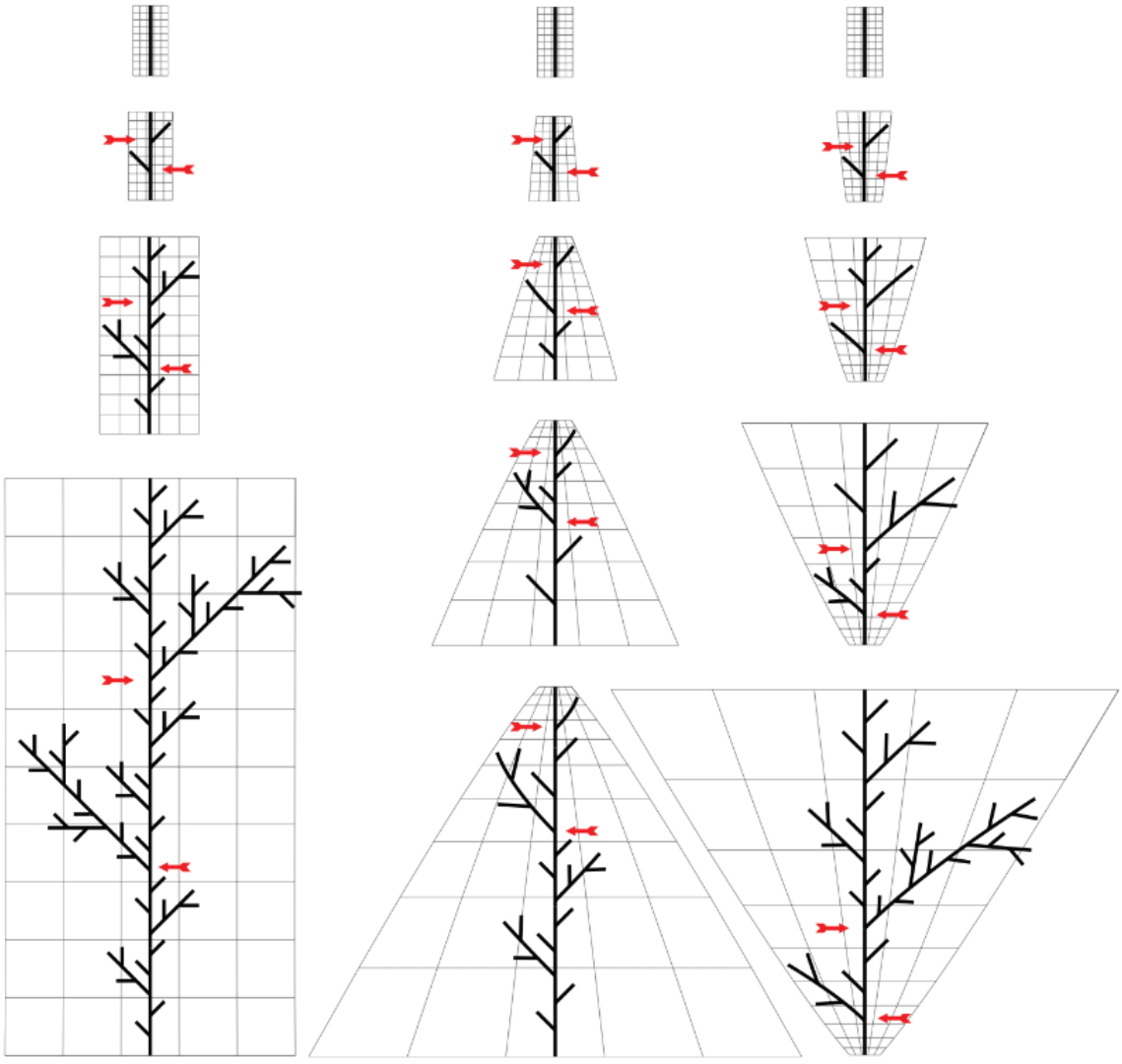

Figure 2.4: Metric driven L system

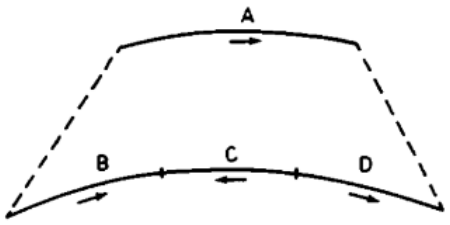

Figure 2.5: Esempio di regola applicata ai lati

gola di produzione consiste nel rimpiazzare ogni ramo che abbia raggiunto un determinato threshold con una struttura ramificata. La differenza tra le colonne sta nella distribuzione di crescita. Nella prima colonna la distribuzione di crescita e' uniforme, dunque tutti i rami raggiungono il threshold simulaneamente e simultaneamente viene applicata la regola di produzione. La colonna 1 e' dunque esattamante cio' che avremmo applicando gli L-systems sopra citata. Nelle colonne 2 e 3 le distribuzioni di crescita sono diverse. Alcuni rami raggiungo il threshold prima degli altri. Nella colonna 2 la crescita e' piu' rapida sopra, nella colonna 1 e' piu' rapida sotto.

2.4 Estensione degli L-systems

In questa sezione vogliamo estendere gli L-system citati sopra per dare una prima approssimazione di un modello metric-driven che tenga conto dello spazio. Questa estensione cosiste nell'estensione delle regole di produzione a spazi 3d.
Il modello che andiamo a descrivere si basa sui concetti di lati (edge), muri (wall) e celle (cell). Le regole di produzione dei lati andranno ad influenzare quelle dei muri, le regole di divisione dei muri influenzano quelle delle celle. Questo concetto sara' poi alla base della riproduzione cellulare e quindi di ogni tipo di morfogenesi.
Indichiamo i lati con lettere maiuscole e consideriamo ad esempio la regola A -> BCB, Questa indica che il lato A sara' diviso in tre segmenti. In questo caso la divisione e' univoca essendo BCB palindromo. Nel caso di regole non palindrome e'necessario specificare un orientamento. Pertanto si usa il '+' per indicare che l'orientamento del nuovo lato e' nella stessa direzione di quello origiario, '-' in caso contrario (Figura 2.5)
In questo caso la orientation rule sara' A-> $B_+C_-D_+$.
Le wall production rule sono invece importanti per definire come i muri di ogni cella devono riprodursi. Quello dei muri e' un concetto importante e altrettanto complesso. Si pensi allo spessore della membrana di una cellula. Questa e' importante per le interazioni cellulari. Data la complessita' dell'argomento, per i propositi di questa sezione, ci si riferira' ai muri come sequenza di lati.
La suddivisione di un muro W_0 segue quella del lati che lo compongono. Le wall

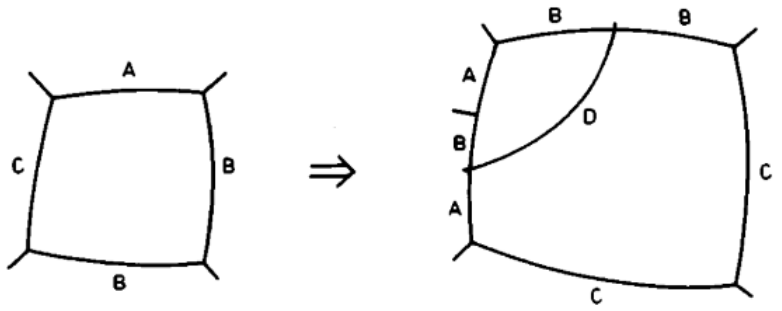

Figure 2.6: Esempio di regola applicata a walls

production rules sono espresse da:
W_0= (S_1 X_1, S_2 X_2, . . ., W_nX_n) dove W_0 e' la sequenza di lati del muro originario, S_i sono i segmenti generati dalle trasformazioni degli edges e gli X_i.
Consideriamo la figura 2.6 e costuiamo ora una wall production rule a partire dai lati.
A -> BB
B -> C
C -> ABA
Al seguito di queste edge production rules, scegliamo come wall production rule: ABBC -> BCCA$\underline{D}$, BAB$\underline{D}$) dove $\underline{D}$ e' il lato aggiunto. Dunque il muro originarario e' stato diviso in due nuovi muri inserendo un nuovo lato $\underline{D}$.
La parte mancante per descrivere la morfogenesi di elementi naturali e' ora la riproduzione delle celle. Una cella dividera' il suo involucro V_0 secondo le regole di produzione dei lati e dei muri. Se una cella si divide in due celle U_1 e U_2 allora un nuovo muro W deve essere creato tra le due.
Piu' formalmente:
V_1= [U_1,W] e V_2= [U_2,W'] dove W' e' il muro W visto da V_2.
La cell production rule sara' allora: V_0 -> ([U_1,W] , [U_2,W'])
Prendiamo ora la figura 2.7 come esempio di applicazione.
Consideriamo le seguenti edge production rules
A -> BB
B -> C
C -> ABA,
le seguenti wall production rules:
ABAB -> (BABA, BABA)
ACAC -> (BCB$\underline{C}$, BCB$\underline{C}$)
BCBC -> ACAC
Cell production rule: [(ABAB)^2, (BCBC)^2, (ACAC)^2] -> [(ABAB)^2, (BCBC)^2, ACAC, $\underline{ACAC}$]^2

Quello che abbiamo appena introdotto e' una generalizzazione degli L-systems a 3 dimensioni. Nei classici L-systems le regole di produzione sono applicate solo o ai lati o ai muri. I sistemi appena descritti non solo sono sviluppati in 3

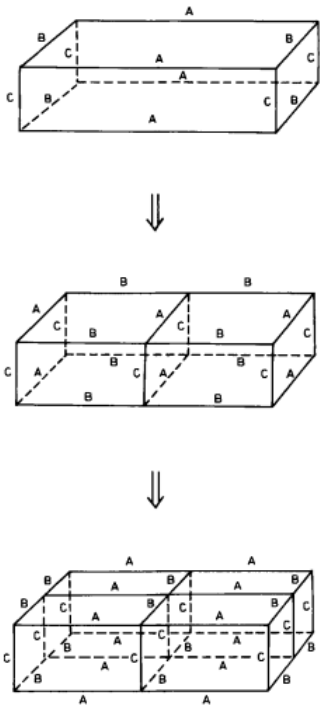

Figure 2.7: Esempio di regola applicata a cells

dimensioni, ma ruotano intorno ai lati a cui vengono ora anche assegnate delle labels. Per questa caretterisca vengon definiti "edge label-controlled cellwork OL-systems"

2.5 Modello variazionale negli schemi di irrigazione

Nelle sezioni precedenti abbiamo dato una definizione topologica di frattali, abbiamo preso in considerazione gli alberi frattali e la loro modelizzazione tramite L-systems e loro generalizzazioni. La domanda a cui vogliamo rispondere ora e': ma perche' questa forma?

I modelli di irrigazione, le piante, le radici hanno una morfologia comune che deriva da ragioni topologiche e di risparmio di energia. Tutti i sistemi citati hanno la forma di alberi che trasportano fluidi da una fonte in tutto il volume.
In questa sezione analizzeremno gli schemi di irrigazione, daremo una forma generica alla loro energia, e vedremo che l'ottimo in termini di costo risulta in schemi da forma frattale.
Andremo a dimostrare che esiste un albero di irrigazione che irriga il volume con resistenza minima.

Questi sistemi di irrigazione sono definiti secondo i seguenti principi:

1- il network irriga l'intero volume di un organismo ed e' richiesta una struttura ramificata gerarchica
2- il network si evolve per minimizzare la dissipazione dell'energia
3- la grandezza del network e' invariante
4- il flusso e' lo stesso per tutto il sistema

Concetto fondamentale e' che per gli scopi di questa sezione, il criterio di minimizzazione dell'energia e' legato alla minimizzazione della resistenza del sistema. L'assunzione di base e' che il sistema abbia una struttura ad albero ramificata e un numero fissato di diramazioni di lunghezza e spessore uniformi (albero omogeneo). Il sistema ha dunque una forma di albero frattale con proprieta' di auto-similarita'.

2.5.1 Costruzione del modello di irrigazione

Prima di procedere con le definizioni formali si vuole dare una idea intuitiva della carrellata di definizioni che seguono, per spiegarne il legame.
Si consiglia, leggendo la parte sottostante, di fare riferimento a questa breve introduzione.
Immaginiamo in tronco di un albero e una serie di fibre che iniziano da una sorgente S e si ramificano con probabilita' p in ogni istante di tempo t, e indichiamole con $X_s(t)$.
Ora tagliamo la sezione del tronco in orizzontale e quello che otterremo sono delle classi di equivalenza. Alla stessa parte di equivalenza apparterranno tutte quelle fibre che si trovano alla stessa altezza in un determinato istante t. Le fibre che appartengono alla stessa classe di equivalenza vengono definite di seguito X-vessel e sono il fondamanento su cui vengono costruite altre definizioni.
L'obiettivo di questo paragrafo e' di formulare una serie di definizioni e di concetti tali da permetterci di affermare che, data una certa funzione di costo, il minimo esiste ed e' un X-vessel, che, per costruzione, ha la forma di albero. Per far questo ci serve dimostrare due concetti, quello di compatezza e di continuita'.
Il concetto base sara' quello dei pattern di irrigazione
Per arrivare a capire cosa e' un pattern di irrigazione verranno elencate una serie di definizioni:
1- il tempo di assorbimento, ossia il tempo dopo il quale un X vessel e' costante
2- X-flow ossia un X-vessel che non ha ancora raggiunto il tempo di assorbimento
3- Spread flow ossia tutti gli X vessel che non presentano fibre che diventano costanti ad un certo istante t
4- Non spread flow, ossia quei flussi che ad un certo punto diventano costanti nel tempo
Terminato l'elenco base di definizioni, si passa alla continuita':
5- Gli X-vessell sono misurabili rispetto alla misura prodotto
6- Per un non spread set di fibre, la convergenza su un compatto implica la con-

vergenza su tutto R_+
7- Se un X flow e' misurabile, si chiamma irrigation pattern
A questo punto si puo' introdurre la funzione di costo I(X) che, se finita, e' tale che gli X siano dei pattern di irrigazione.
Se una successione $X - n$ appartenente ai pattern di irrigazione converge ad una X, allora quella X appartiene ai pattern di irrigazione.
Per terminare l'excursus sulla continuita', si introduce il concetto di misura di irrigazione e si puo' dimostrare che che i pattern di irrigazione convergono debolmente rispetto alla misura di irrigazione.
Il concetto mancante e' ora quello della compattezza. L'insieme dei pattern di irrigazione non e' un compatto, ma si puo' dimistrare che esiste sempre un pattern di irrigazione avente la forma di un istogramma che e' equivalente a al pattern di irrigazione stesso. Quindi questo e' sufficiente per avere la prorpieta' di compatezza che cercavamo.
Abbiamo dunque sia la continuita' che la compatezza che sono le due proprieta' che cercavamo per poter affermare l'esistenza e l'unicita' del minimo della funzione di costo.

Di seguito sono riportati i dettagli formali.

Sia (Ω, |.|) uno spazio di probabilita' che interpretiamo come un tronco di un albero formato da fibre.
Un set di fibre di Ω e' una funzione:

$$X : \Omega x \mathbb{R}^+ - > \mathbb{R}^N \tag{2.5.1}$$

tale che:
1- Per ogni $p \in \Omega$, $X_p(t)$: t -> $X(p,t)$ e' - continua.
2- Per ogni $p \in \Omega$, $X_p(0)$=S

Data la fonte $S \in \mathbb{R}^N$, definiamo $C_S(\Omega)$ il set di tutte le X di Ω aventi S come fonte. Il tempo t ha un significato geometrico e indica la distanza dalla fonte.
Definizione1:
Data $t \in \mathbb{R}^+$, diciamo che due punti p e q in Ω appartengono allo stesso X-vessel se sono equivalenti rispetto a t, ossia se

$$X_p(s) = X_q(s) \text{per ogni s} \in [0,t] \tag{2.5.2}$$

e indicheremo $p \simeq q$.
In altre parole e' come se stessimo sezionando i vari flussi orizzontalmente. E tutti i flussi che soddisfano la 2.5.2 appartengono alla stessa classe di equivalenza di p indicata con $[p]_t$ e nominata X-vessel di p al tempo t.

Definizione2:

Sia $X \in C_s(\Omega)$. Definiamo tempo di assorbimento la funzione σ_X: $\Omega \to \mathbb{R}_N$

tale che

$$\sigma_X(p) := inf\{t \in \mathbb{R}^+ | X_p(s) \text{ e' costante su } [t + \infty]\} \tag{2.5.3}$$

Un punto $p \in \Omega$ e' assorbito se $\sigma_X(p) < \infty$. Un punto p e' assorbito in t se $\sigma_X(p) \leq t$.

Definizione3:

Sia $X \in C_S(X)$ e t>0.Diciamo che $X \in \Omega$ e' un set di assorbimento se al tempo t se $\sigma_X(p) \leq t$ per ogni $p \in X$. Se X $\in$ appartiene all'insieme delle classi di X $V_t(X)$ allora sara' definito X-vessell assorbito. Definiamo $A_t()$ l'insieme dei punti assorbiti al tempo t.

Definizione4:

Sia $X \in C_S(X)$ e t>0 e $X \in V_t(X)$. Diciamo che X e' un X-flow a tempo t se non e' un absorbed X-vessel. Definiamo con $F_t(X)$ l'insieme di X-flows al tempo t.

Proposizione

Se $X \in C_S(X)$ e' una funzione misurabile rispetto alla misura prodotto, allora tutti gli X-vessel di $V_t(X)$ sono misurabili.

Proposizione

Sia $X \in C_S(X)$ un non-spread set di fibre. Sia $D \in \mathbb{R}^+$ tale che per ogni $t \in D$, ogni $X - flow$ in $F_t(X)$ sia misurabile. Allora $X_t = X(., t)$ e' una funzione misurabile per ogni $t \in R$.

Lemma
Sia $f : \Omega X \mathbb{R}_+ \to \mathbb{R}$ tale che $f(., t)$ sia misurabile per t in $D \in R_+$ e $f(p, .)$ sia continua per ogni $p \in \Omega$. Allora f e' una funzione misurabile.

Proposizione
Sia $X \in C_S(\omega)$ un non-spread set di fibre. Allora X e' misurabile se e sono se per ogni t appartenente a D ogni X-flow $V \in F_t(X)$ e' un insieme misurabole e se e solo se per ogni $t \in \mathbb{R}_+$ ogni X-vessel $V \in V_t(X)$ e' un set misurabile.

Teorema:
Per ogni set di fibre $X \in C_S(X\Omega$ sono equivalenti le seguenti proposizioni:
1- X e' misurabile

2- $X(.,t)$ e' misurabile per ogni $t \in D$
3- $X(.,t)$ e' misurabile per ogni $t \in \mathbb{R}_+$

Inoltre se X e' un non- spread set di fibre, anche le seguenti definizioni sono equivalenti tra loro e alle precedenti.
4- Ogni $X - vessel$ e' misurabile per ogni $t \in \mathbb{R}_+$
5- Ogni X-flow e' misurabile per ogni $t \in D$

Definizione
Diciamo che $X_i nC_S(\Omega)$ e' un irrigation pattern di Ω se ' misurabile. L'insieme degli irrigation patter di Ω si denota con $P_S(\Omega)$.

Proposizione
Per ogni $X \in P_S(\Omega)$, funzione di assorbimento σ_X e' una funzione misurabile

Proposizione
Se $X \in P_S(\Omega)$ e' una non spread funzione di irrigazione, allora per ogmi $p \in \Omega$ e per ogni $t < \sigma_X(p)$, esiste $V \in F_t(X)$ tale che $p \in V$

2.5.2 Funzione di Costo e misura di irrigazione

Definiamo la funzione di costo come

$$c_x(t) = \int_{\mathbb{R}_+} c_x(t)\, dt \tag{2.5.4}$$

dove

$$I(X) = \int_{M_t(X)} \varphi_X(t)\, dp \tag{2.5.5}$$

e

$$\varphi_X(p,t) : (p,t) \to [[p_t]]^{\alpha-1} \tag{2.5.6}$$

con α in (-1 1) e $X \in P_S(\Omega)$.
Definiamo la misura di irrigazione come:

$$\mu_X(A) = i_X^{-1}(A) \tag{2.5.7}$$

dove

$$i_X(p) = X(p, \sigma_X(p)) \tag{2.5.8}$$

definito sull'insieme dei punti di assorbimento A_X.

Teorema:
Sia X_n, $n \in N$ una seguenza di pattern di irrigazione tali che $I(X_n) < \infty$. Se $X - n \to X$ allora $\mu_{Xn} \to \mu_X$

2.5.3 Equivalenza dei pattern di irrigazione e proprieta' di semi continuita'

Definizione:
Siano Ω e Ω_0 due spazi di probabilita'. Consideriamo due pattern di irrigazione $X \in P_S\Omega$ e $X_0 \in P_S(\Omega)$. Diciamo che X e X_0 sono equivalenti se per ogni $t \in D$ sottoinsieme denso di $\mathbb{R}_+$ esiste una biezione ϕ^t: $F_t(X) \to F_t(X_0)$ tale che per ogni $V \in F_t(X)$, $X_V = X'\phi^t(V)$

Teorema
Se $X_n \to X$ allora

$$i(X) \leq \liminf I(X_n) \tag{2.5.9}$$

Proprieta' di compattezza

Sia X_n una successione di istogrammi con $I_X(n)$ limitata. Allora esiste sempre una sotto-successione di istrogrammi X_{kn} che converge ad un pattern di irrigazione X.

2.5.4 Minimizzazione del problema di irrigazione

Gazie alla proprieta' di copattezza, alla semicontuinita', alla possibilita' di minimizare pattern di irrigazione tramite istogrammi, e' possibile trovare il minimo della seguente funzione di costo:

$$\min_X E(X) \tag{2.5.10}$$

dove $XinP_S(\Omega)$ e $\Omega \in [0\ 1]$ e E(X) e' un funzionale definito da:

$$E(X) = I(X) + J(\mu_X) \tag{2.5.11}$$

dove J e' un funzionale definito da:

$$J : M^+(\mathbb{R}_\mathbb{N}) \to R \tag{2.5.12}$$

)

.

CAPITOLO 3

Applicazioni

Questa sezione e' una applicazione informatica di quanto descritto nei capitoli precedenti. L'idea e' di generare alberi frattali semplici e poi applicare funzioni in Java script che ne generano rotazioni parziali. Questo Capitolo ha come fonti [8] e [3]

Il seguente algoritmo e' il "main" script utilizzato per richiamare le funzioni di rotazione o di semplice rappresentazione grafica dei frattali ed e' quello utilizzato per far girare gli algoritmi che verranno di seguito descritti.

```
<!DOCTYPE html>
<html>
<head>
    <title></title>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery
       /2.1.3/jquery.min.js"></script>
    <script type="text/javascript" src="pytreeanim.js"></
       script>
    <style type="text/css">
    html, body {
        margin: 0px;
    }

    canvas {
        display: block;
        position: absolute;
        top: 0px;
        left: 0px;
    }

```

```
    </style>
</head>
<body>
<canvas id="canvas"></canvas>
</body>
</html>
```

Main

La funzione di Main e' la base che serve per costruire un albero. Partiamo da due punti $p0$ e $p1$ definiti, un angolo random tra 0 e $\pi/2$ e construiamo una funzione ricorsiva che ad ogni passi trasli i due punti iniziali di branchAngleA e branchAngleB, e poi applichi una rotazione.

```
window.onload = function() {
    var canvas = document.getElementById("canvas"),
        context = canvas.getContext("2d"),
        width = canvas.width = window.innerWidth,
        height = canvas.height = window.innerHeight;

    var p0 = {
            x: width / 2,
            y: height - 50
        },
        p1 = {
            x: width / 2,
            y: 50
        },
        branchAngleA = randomRange(-Math.PI / 2, Math.PI / 2),
        branchAngleB = randomRange(-Math.PI / 2, Math.PI / 2),
        trunkRatio = randomRange(0.25, 0.75);

    function randomRange(min, max) {
        return min + Math.random() * (max - min);
    }

    tree(p0, p1, 8);

    function tree(p0, p1, limit) {
        var dx = p1.x - p0.x,
            dy = p1.y - p0.y,
            dist = Math.sqrt(dx * dx + dy * dy),
            angle = Math.atan2(dy, dx),
            branchLength = dist * (1 - trunkRatio),
            pA = {
                x: p0.x + dx * trunkRatio,
                y: p0.y + dy * trunkRatio
```

```
59              },
60              pB = {
61                  x: pA.x + Math.cos(angle + branchAngleA) *
                       branchLength,
62                  y: pA.y + Math.sin(angle + branchAngleA) *
                       branchLength,
63              },
64              pC = {
65                  x: pA.x + Math.cos(angle + branchAngleB) *
                       branchLength,
66                  y: pA.y + Math.sin(angle + branchAngleB) *
                       branchLength,
67              };
68
69          context.beginPath();
70          context.moveTo(p0.x, p0.y);
71          context.lineTo(pA.x, pA.y);
72          context.stroke();
73
74          if(limit > 0) {
75              tree(pA, pC, limit - 1);
76              tree(pA, pB, limit - 1);
77          }
78          else {
79              context.beginPath();
80              context.moveTo(pB.x, pB.y);
81              context.lineTo(pA.x, pA.y);
82              context.lineTo(pC.x, pC.y);
83              context.stroke();
84          }
85          branchAngleA += randomRange(-0.02, 0.02);
86          branchAngleB += randomRange(-0.02, 0.02);
87          trunkRatio += randomRange(-0.02, 0.02);
88      }
89  };
```

Pytree

La seguente funzione genera un albero a partire da un quadrato. Sul lato superiore costruisce un triangolo e su ciascun lato un quadrato e cosi' via

Ecco di seguito il risultato grafico:

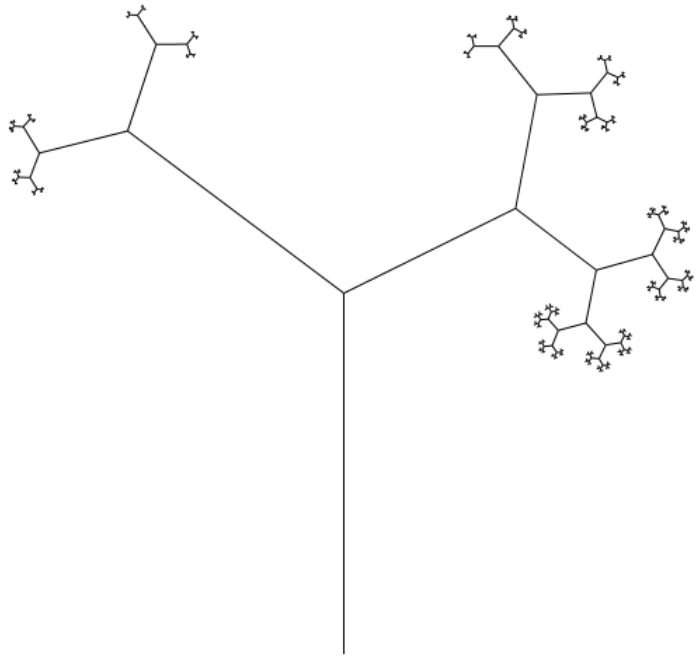

Figure 3.1: Albero

```
90  window.onload = function() {
91      var canvas = document.getElementById("canvas"),
92          context = canvas.getContext("2d"),
93          width = canvas.width = window.innerWidth,
94          height = canvas.height = window.innerHeight;
95
96      var branchAngleA = 0,
97          t = 0;
98
99      draw();
100
101     function draw() {
102         context.clearRect(0, 0, width, height);
103         branchAngleA = -Math.PI / 4 + Math.sin(t += 0.01) *
```

```
            Math.PI / 4;
104         tree(width / 2 - 75, height, 150, 0, 8);
105         requestAnimationFrame(draw);
106     }
107
108
109     function randomRange(min, max) {
110         return min + Math.random() * (max - min);
111     }
112
113     function tree(x, y, size, angle, limit) {
114         context.save();
115         context.translate(x, y);
116         context.rotate(angle);
117         context.fillRect(0, 0, size, -size);
118         contetx.beginPath();
119         contetx.fillStyle = "red";
120        contex.arc(x/2, y/2, size/2, 0, 2 * Math.PI);
121        contetx.fillStyle = "black";
122        // contex.stroke();
123
124         // left branch
125         var x0 = 0,
126             y0 = -size,
127             size0 = Math.abs(Math.cos(branchAngleA) * size),
128             angle0 = branchAngleA;
129
130         if(limit > 0) {
131             tree(x0, y0, size0, angle0, limit - 1);
132         }
133         else {
134             context.save();
135             context.translate(x0, y0);
136             context.rotate(angle0);
137             context.fillRect(0, 0, size0, -size0);
138             contetx.beginPath();
139             contetx.fillStyle = "red";
140             contex.arc(x0/2, y0/2, size0/2, 0, 2 * Math.PI);
141             contetx.fillStyle = "black";
142             // contex.stroke();
143
144             context.restore();
145         }
146
147         // right branch
148         var x1 = x0 + Math.cos(angle0) * size0,
149             y1 = y0 + Math.sin(angle0) * size0,
```

```
150             size1 = Math.abs(Math.sin(branchAngleA) * size)
151             angle1 = angle0 + Math.PI / 2;
152
153         if(limit > 0) {
154             tree(x1, y1, size1, angle1, limit - 1);
155         }
156         else {
157             context.save();
158             context.translate(x1, y1);
159             context.rotate(angle1);
160             context.fillRect(0, 0, size1, -size1);
161             context.restore();
162         }
163
164
165         context.restore();
166     }
167 };
```

3.0.1 Albero animato

Il seguente algoritmo e' una animazione dell' algoritmo precedenet:

```
168 window.onload = function() {
169     var canvas = document.getElementById("canvas"),
170         context = canvas.getContext("2d"),
171         width = canvas.width = window.innerWidth,
172         height = canvas.height = window.innerHeight;
173
174     var branchAngleA = 0,
175         t = 0;
176
177     draw();
178
179     function draw() {
180         context.clearRect(0, 0, width, height);
181         branchAngleA = -Math.PI / 4 + Math.sin(t += 0.01) *
               Math.PI / 4;
182         tree(width / 2 - 75, height, 150, 0, 8);
183         requestAnimationFrame(draw);
184     }
185
186
187     function randomRange(min, max) {
188         return min + Math.random() * (max - min);
189     }
190
```

```
191     function tree(x, y, size, angle, limit) {
192         context.save();
193         context.translate(x, y);
194         context.rotate(angle);
195         context.fillRect(0, 0, size, -size);
196 
197         // left branch
198         var x0 = 0,
199             y0 = -size,
200             size0 = Math.abs(Math.cos(branchAngleA) * size),
201             angle0 = branchAngleA;
202 
203         if(limit > 0) {
204             tree(x0, y0, size0, angle0, limit - 1);
205         }
206         else {
207             context.save();
208             context.translate(x0, y0);
209             context.rotate(angle0);
210             context.fillRect(0, 0, size0, -size0);
211             context.restore();
212         }
213 
214         // right branch
215         var x1 = x0 + Math.cos(angle0) * size0,
216             y1 = y0 + Math.sin(angle0) * size0,
217             size1 = Math.abs(Math.sin(branchAngleA) * size)
218             angle1 = angle0 + Math.PI / 2;
219 
220         if(limit > 0) {
221             tree(x1, y1, size1, angle1, limit - 1);
222         }
223         else {
224             context.save();
225             context.translate(x1, y1);
226             context.rotate(angle1);
227             context.fillRect(0, 0, size1, -size1);
228             context.restore();
229         }
230 
231 
232         context.restore();
233     }
234 };
```

Ecco di seguito il risultato grafico:

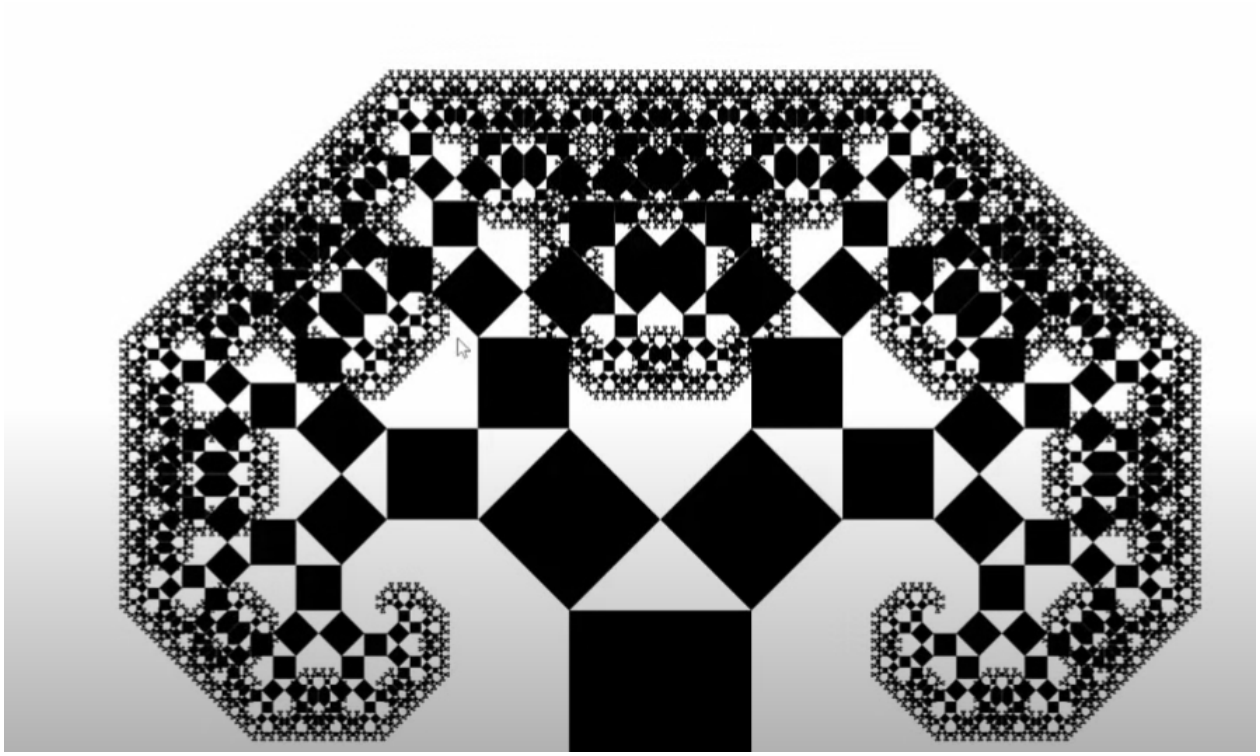

Figure 3.2: Albero Animato

3.0.2 Applicazione musicale

```
238 package music;
239 import org.jfugue.player.Player;
240 import java.lang.Math;
241 import java.awt.Color;
242 import java.awt.Graphics;
243 import java.awt.Graphics2D;
244 import java.awt.image.BufferedImage;
245 import java.io.File;
246 import java.io.IOException;
247 import java.util.ArrayList;
248 import java.util.HashMap;
249 import java.util.Map;
250 import java.util.Random;
251
252 import org.jfugue.midi.MidiFileManager;
253 import org.jfugue.pattern.Pattern;
254 import org.jfugue.theory.Chord;
255 import org.jfugue.theory.ChordProgression;
256 import org.jfugue.theory.Note;
257 import org.staccato.ReplacementMapPreprocessor;
258
```

```
259 import javax.swing.*;
260
261 import java.lang.Math;
262
263 import java.util.logging.Logger;
264
265 public class MyMusic {
266
267     public static final int WIDTH=800;
268     public static final int HEIGHT=600;
269     public static final int ITERATIONS=4;
270     public static final int SCALE=200;
271     public static int MODULO = 12;
272     public static final int TEMPI = 16;
273     static ArrayList<String> punti = new ArrayList<String>();
274     public static final String FRACTAL = "Fractal_tree";
275
276     @SuppressWarnings("static-access")
277     public static void main(String[] args) throws IOException
        {
278
279
280         int JUMP = 20;
281         int len = 20000;
282         System.out.println("Start "+ FRACTAL + " song");
283         if(FRACTAL.compareTo("Fractal_tree")==0) {
284             JUMP = 43;
285             len = 20000;
286             for(int x=0;x<WIDTH;x++){
287                 for(int y=0;y<HEIGHT;y++) {
288                     ArrayList<String> newpoints =
                        calculatePointDinamicTree
                        (2,3,50,10,100);
289                     if(newpoints != null) {
290                         punti.addAll(newpoints);
291                     }else {
292                         break;
293                     }
294
295                 }
296             }
297         }
298
299
300
301
302
```

```
303         System.out.println("\nPunti array lenght: "+punti.size
                ());
304         System.out.println("Inizio calcolo note e durata");
305         String note = "";
306 
307 
308         for (int i = 1;i<len-JUMP;i=i+JUMP) {
309             String[] p = punti.get(i).split("_");
310             float x = Float.parseFloat(p[0]);
311             float y = Float.parseFloat(p[1]);
312             int nota = abs((int) (x%MODULO));
313             int ottava =abs((int)y%10);
314 
315             System.out.println("Nota :"+nota+" "+ottava);
316 
317             String[] p1 = punti.get(i+JUMP).split("_");
318             float x1 = Float.parseFloat(p1[0]);
319             float y1 = Float.parseFloat(p1[1]);
320 
321             float distanza = (float) Math.sqrt(Math.pow(Math.
                    abs(x1-x), 2)+Math.pow(Math.abs(y1-y), 2));
322 
323             System.out.println("Distanza: "+ distanza);
324 
325             //calcolo il modulo 16 della distanza, cos  al
                    massimo ho delle semicrome -> 0 sta per 1
326 
327             int durata = (int) (distanza%TEMPI);
328             System.out.println("Durata: "+ durata+"/16");
329             String nota_str = makeNote(nota)+ottava+
                    makeDuration(durata);
330 
331             System.out.println("Nota finale: "+nota_str);
332 
333 
334 
335             if(i==0) {
336                 note = nota_str;
337             } else {
338                 note += " "+nota_str;
339             }
340 
341             System.out.println("-------------------------");
342 
343 
344         }
345 
```

```
346         //System.out.println("Stringa note:\n"+ note);
347
348
349         Player player = new Player();
350         Pattern axiom = new Pattern("T110 " + "V0 I[Piano] "+
                note);
351         MidiFileManager mfm = new MidiFileManager();
352         File f = new File(FRACTAL+"_v1.midi");
353         mfm.save(player.getSequence(axiom), f);
354
355         player.play(axiom);
356
357
358
359         System.out.println("************************End
                ************************");
360     }
361
362     public static String makeNote(int nota) {
363         switch(nota) {
364         case 0:
365             return "C";
366         case 1:
367             return "C#";
368         case 2:
369             return "D";
370         case 3:
371             return "D#";
372         case 4:
373             return "E";
374         case 5:
375             return "F";
376         case 6:
377             return "F#";
378         case 7:
379             return "G";
380         case 8:
381             return "G#";
382         case 9:
383             return "A";
384         case 10:
385             return "A#";
386         case 11:
387             return "B";
388         }
389         return "";
390     }
```

```
391
392         public static String makeDuration(int nota) {
393             switch(nota) {
394             case 0:
395                 return "w";
396             case 1:
397                 return "s";
398             case 2:
399                 return "i";
400             case 3:
401                 return "sss";
402             case 4:
403                 return "q";
404             case 5:
405                 return "sssss";
406             case 6:
407                 return "iii";
408             case 7:
409                 return "s7";
410             case 8:
411                 return "h";
412             case 9:
413                 return "s9";
414             case 10:
415                 return "i5";
416             case 11:
417                 return "s11";
418             case 12:
419                 return "q3";
420             case 13:
421                 return "s13";
422             case 14:
423                 return "i7";
424             case 15:
425                 return "s15";
426             }
427
428
429         return "";
430     }
431
432     public static int abs(int a) {
433         if(a>=0)
434             return a;
435         return -a;
436     }
437
```

```
438     public static ArrayList<String> calculatePoint(float x,
           float y) {
439         float cx=x;
440         float cy=y;
441         int i=0;
442         ArrayList<String> array = new ArrayList<String>();
443 
444         for(; i<ITERATIONS;i++) {
445             float nx= x*x -y*y+cx;
446             float ny = 2*x*y+cy;
447             x=nx;
448             y=ny;
449 
450             if(x*x +y*y < 4) {
451                 array.add(new String(x+"_"+y));
452                 //System.out.print("\nadded "+x +" "+y);
453             };
454 
455         }
456         if(i==ITERATIONS) {
457         return array;
458         }
459         return array;
460     }
461 
462     public static String wordFractal(int n) {
463         if (n < 2)
464             return n == 1 ? "1" : "";
465 
466         // we should really reserve fib n space here
467         StringBuilder f1 = new StringBuilder("1");
468         StringBuilder f2 = new StringBuilder("0");
469 
470         for (n = n - 2; n > 0; n--) {
471             String tmp = f2.toString();
472             f2.append(f1);
473 
474             f1.setLength(0);
475             f1.append(tmp);
476         }
477 
478         return f2.toString();
479     }
480 
481     public static ArrayList<String> pointWordFractal(int x,
           int y, int dx, int dy, int N) {
482         String word = wordFractal(N);
```

```
483         ArrayList<String> lista = new ArrayList<String>();
484         for (int n = 0; n < word.length(); n++) {
485 
486             lista.add(x+"_"+y);
487             x += dx;
488             y += dy;
489             if (word.charAt(n) == '0') {
490                 int tx = dx;
491                 dx = (n % 2 == 0) ? -dy : dy;
492                 dy = (n % 2 == 0) ? tx : -tx;
493             }
494         }
495         return lista;
496     }
497 
498     public static void pointTree(int x1, int y1, double angle,
        int depth) {
499         if (depth == 0) return;
500         int x2 = x1 + (int) (Math.cos(Math.toRadians(angle)) *
            depth * 10.0);
501         int y2 = y1 + (int) (Math.sin(Math.toRadians(angle)) *
            depth * 10.0);
502         punti.add(x2+"_"+y2);
503         pointTree( x2, y2, angle - 20, depth - 1);
504         pointTree( x2, y2, angle + 20, depth - 1);
505         return;
506 
507     }
508 
509     public static void pointCantor(int start, int depth, int
       index) {
510         int seg = depth/3;
511 
512         if (seg == 0) return;
513         for (int i = index; i < HEIGHT; i++) {
514             for (int j = start; j < start + seg; j++) {
515                 Random r = new Random();
516                 r.setSeed(3);
517                 int low = 0;
518                 int high = 11;
519                 int result = r.nextInt(high-low) + low;
520                 punti.add(j+"_"+result);
521             }
522         }
523         pointCantor(start, seg, index + 1);
524         pointCantor(start + seg * 2, seg, index + 1);
525 
```

```
526         return;
527     }
528 
529     public static ArrayList<String> calculatePointDinamicTree(
        float p0X, float p1X,float p0Y, float p1Y, float limit)
         {
530 
531         double branchAngleA = Math.PI/4;
532         double branchAngleB =  Math.PI/6;
533         double trunkRatio= 0.6;
534 
535         ArrayList<String> array = new ArrayList<String>();
536 
537                 float dx = p1X - p0X;
538                 float dy = p1Y - p0Y;
539                 float dist = (float) Math.sqrt(dx * dx + dy *
                    dy);
540                 float angle = (float) Math.atan2(dy, dx);
541                  float branchLength = (float) (dist * (1 -
                     trunkRatio));
542 
543                  float pAX= (float) (p0X + dx*trunkRatio);
544                  float pAY= (float) (p0Y + dy*trunkRatio);
545 
546                  array.add(pAX+"_"+pAY);
547 
548                  float pBX= (float) (pAX + Math.cos(angle +
                     branchAngleA) * branchLength);
549                  float pBY= (float) (pAY + Math.sin(angle +
                     branchAngleA) * branchLength);
550 
551                  array.add(pBX+"_"+pBY);
552 
553                  float pCX= (float) (pAX + Math.cos(angle +
                     branchAngleB) * branchLength);
554                  float pCY= (float) (pAY + Math.sin(angle +
                     branchAngleB) * branchLength);
555 
556                  array.add(pCX+"_"+pCY);
557 
558             if(limit > 0) {
559                 calculatePointDinamicTree(pAX,pCX, pAY,pCY,
                    limit - 1);
560                 calculatePointDinamicTree(pAX,pBX, pAY,pBY,
                    limit - 1);
561             }
562 
```

```
563                 else  return array;
564                 return array;
565         }
566
567
568
569 }
```

Bibliografia

[1] Gerald Edgar. *Measure, topology, and fractal geometry*. Springer Science & Business Media, 2007.

[2] Kenneth Falconer. *Fractal geometry: mathematical foundations and applications*. John Wiley & Sons, 2004.

[3] Maria Antonia Cupi. Gli oggetti frattali: dalla geometria alla musica= fractal objects: from geometry to music. Master's thesis, Politecnico di Torino, 2020.

[4] Wikipedia. L-system [online] available: https://www.dropbox.com/s/b3ckaxxvhg5of97/072021.

[5] Przemyslaw Prusinkiewicz, Brendan Lane, and Adam Runions. Metric-driven grammars and morphogenesis. In *Conference on Computability in Europe*, pages 347–351. Springer, 2014.

[6] Francesco Maddalena, Sergio Solimini, and Jean-Michel Morel. A variational model of irrigation patterns. *Interfaces and Free Boundaries*, 5(4):391–416, 2003.

[7] Aristid Lindenmayer. Models for plant tissue development with cell division orientation regulated by preprophase bands of microtubules. *Differentiation*, 26(1-3):1–10, 1984.

[8] CodingMath. Codingmath [online] available: https://github.com/bit101/codingmath/tree/master/episode40, 2015.

[9] Wikipedia. L-system [online] available: https://it.wikipedia.org/wiki/sistema$_d i_l indenmayer$, 2021.

[10] rosettaCode. Codingmath [online] available: https://rosettacode.org/wiki/fractal$_t ree$, 2020.