UNIVERSITAT DE BARCELONA
Facultat de Matemàtiques i Informàtica

GRAU DE MATEMÀTIQUES

Treball final de grau

ANÀLISI I SIMULACIÓ D'ARBRES FRACTALS

Autor: Oscar Romero Moreno

Director: Dr. Antoni Benseny Ardiaca
Realitzat a: Departament de Matemàtiques i Informàtica

Barcelona, 17 de juny de 2020

Abstract

Fractal geometry is able to accurately describe many of the irregular and fragmented shapes that surround us in nature. This work is focused on deterministic fractals generated from iterated function systems, especially in fractal trees. The software created allows to generate fractals, observe the main features and simulate elements of nature.

Resum

La geometria fractal és capaç de descriure amb precisió moltes de les formes irregulars i fragmentades que ens envolten a la natura. Aquest treball se centra en els fractals deterministes generats a partir de sistemes iteratius de funcions, especialment en els arbres fractals. Els programes creats permeten generar fractals, observar les principals característiques i simular elements de la natura.

2010 Mathematics Subject Classification. 28A80

Índex

1 La geometria fractal de la natura

L'any 1982, el matemàtic Benoît Mandelbrot publica *Fractal geometry of nature* i en la introducció fa la reflexió següent:

Per què sovint es descriu la geometria com una cosa freda i àrida? Una de les raons és per la seva incapacitat de descriure la forma d'un núvol, una muntanya, una costa, un arbre o un raig. Doncs perquè ni els núvols són esfèrics, ni les muntanyes còniques, ni les costes circulars, ni el tronc d'un arbre cilíndric, ni un raig rectilini.

Amb l'objectiu d'estudiar aquestes formes, indescriptibles de forma acurada només amb la geometria clàssica, Mandelbrot va desenvolupar una nova geometria: la geometria fractal. Com s'ha comentat anteriorment, aquesta permet descriure amb precisió moltes de les formes irregulars i fragmentades que ens envolten a la natura com els exemples abans donats i gràcies a això s'ha convertit en poques dècades en una eina multidisciplinària utilitzada per científics, metges, artistes, informàtics o meteoròlegs entre altres.

Tot i que la definició formal es donarà més endavant en el treball, podem definir un fractal com un model matemàtic o un objecte real que repeteix la seva estructura bàsica, fragmentada o aparentment irregular, en diferents escales. Aquesta característica es coneix com autosimilitud. Podem diferenciar dos tipus de fractals: els fractals artificials i els fractals naturals. D'una banda els fractals artificials són aquells models matemàtics autosimilars en infinites escales, amb exemples típics com la corva de Koch, el triangle de Sierpinski o el conjunt de Cantor. D'altra, els fractals naturals són aquells objectes reals que es poden descriure de forma aproximada i estadística mitjançant la geometria fractal però la seva "autosimilitud" s'extén només en un cert rang d'escales, amb exemples de la natura com els descrits abans.

I justament la curiositat per la gran quantitat d'aquests exemples que es poden observar a la natura i les ganes de conèixer amb més profunditat les matemàtiques amagades són el motor d'aquesta memòria. El propòsit d'aquest treball és fer un estudi de la geometria fractal que s'amaga en una part concreta de la natura: la ramificació de plantes i arbres. Se centrarà l'atenció en un tipus concret de fractals que s'anomenen arbres fractals o *fractal trees* en anglès. Per fer-ho, caldrà primer un estudi dels fractals artificials que ens servirà de base per al posterior estudi dels fractals naturals amb l'ajuda de programari per crear-los.

Servint de motivació, donem l'exemple següent. Suposem que volem definir l'estructura de branques d'un arbre. Difícilment podem definir-la amb la geometria clàssica perquè la quantitat de branques sovint és molt elevada i l'organització complicada. En canvi, si pensem en definir-la en termes de "la relació entre les peces", podem veure que la tasca és molt més senzilla. Suposem que tenim una branca inicial i que al final d'aquesta es creen tres branques de mida la meitat que la inicial en un cert angle. De nou, al final de cada branca es tornen a crear tres noves branques de mida la meitat que l'anterior en el mateix angle. Repetint el procés una sèrie de vegades, es forma una estructura curiosament similar a l'estructura de branques d'un arbre com la següent:

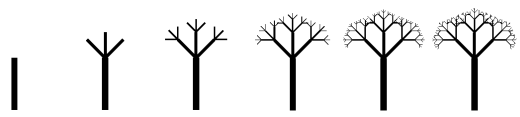

Figura 1: Generació d'un arbre fractal.

Estructura de la memòria

L'estructura de la memòria es divideix en tres parts principals: l'estudi teòric dels fractals, la introducció al programari utilitzat i l'explicació dels diferents programes amb què es generaran fractals.

L'estudi dels fractals artificials es duu a terme en l'apartat 2 de la memòria. Es comença definint l'espai mètric on viuen els fractals i demostrant la seva completesa. A continuació, es defineixen conceptes com les transformacions i les aplicacions contractives que serveixen de base per definir els sistemes iteratius de funcions o IFS, moment en el qual es dona una definició rigorosa de fractal determinista. Seguidament es presenten els dos algoritmes per calcular fractals a partir d'IFS i els conjunts de condensació, que serveixen per crear els arbres fractals. Per últim, es demostra el Teorema del Collage i la dependència contínua dels fractals en els paràmetres.

L'apartat 3 centra l'atenció en el programari utilitzat per visualitzar fractals. Primerament es mostra com es dibuixen objectes amb les biblioteques utilitzades, després es fa una explicació de les matemàtiques que fan de base del funcionament dels diferents programes i, per acabar, s'analitza l'estructura general d'aquests i s'introdueix l'objectiu de cadascun.

En els quatre apartats següents, del 4 al 7, es comenten els principals trets dels quatre programes creats. Els dos primers permeten generar fractals artificials pels dos algoritmes que es presenten a la part teòrica, el tercer genera arbres fractals dos-dimensionals utilitzant un dels algoritmes anteriors i l'últim permet crear arbres fractals tres-dimensionals que simulen fractals naturals.

Als annexos es poden trobar alguns conceptes fonamentals que es donen per coneguts a l'apartat 2, a més d'una petita explicació sobre la importància de la il·luminació i les textures en les visualitzacions 3D.

2 L'estudi dels fractals artificials

Com s'ha dit abans, els fractals artificials són aquells objectes matemàtics que mantenen la seva forma essencial per infinits canvis d'escala. En aquest apartat ens marquem l'objectiu de donar una definició rigurosa dels fractals artificials i trobar les principals propietats i característiques, centrant-nos principalment en els fractals deterministes o creats sense aleatorietat. Per fer-ho, que millor que primer observar i sobretot apreciar alguns exemples...

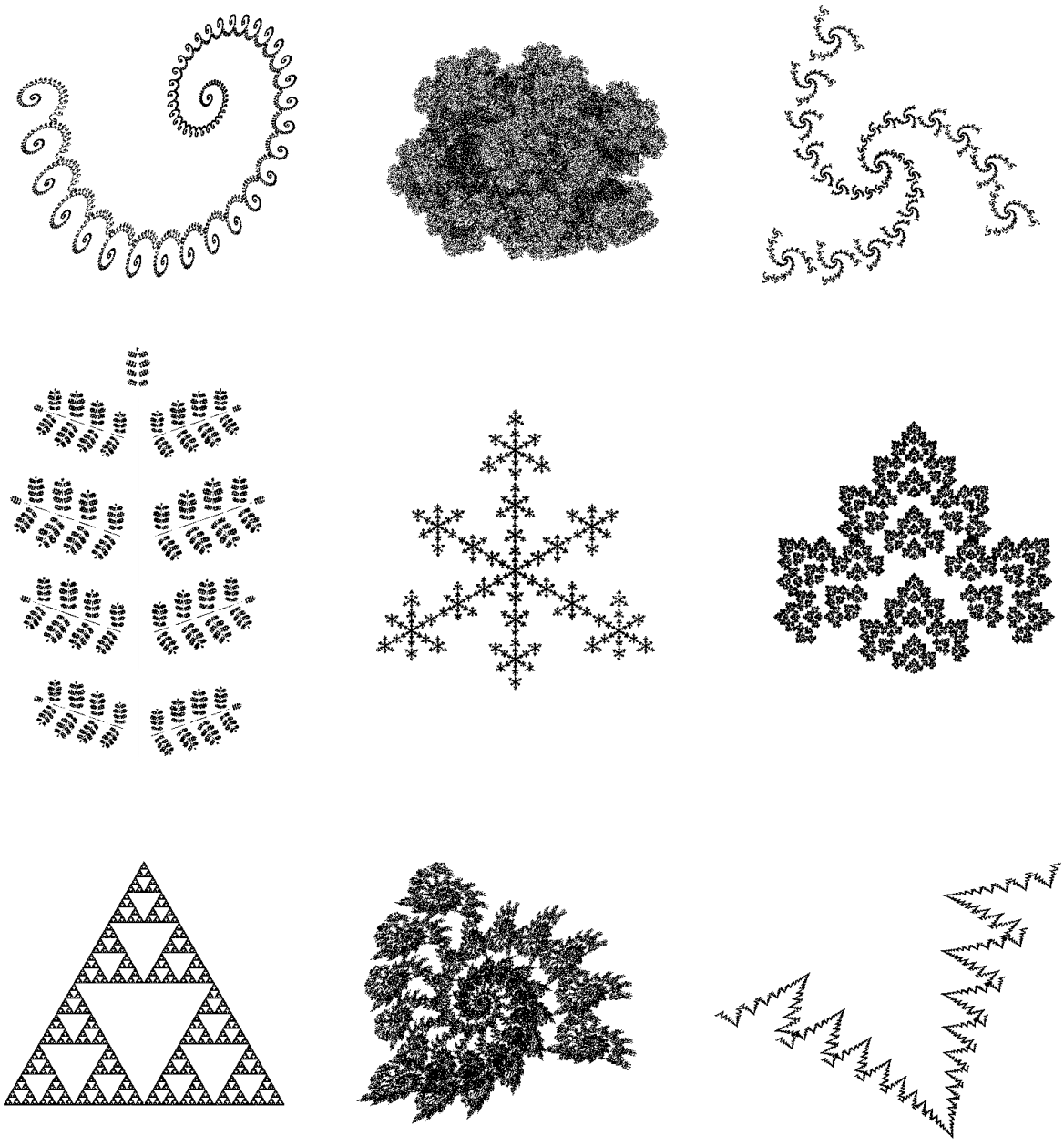

Figura 2: Exemples de fractals.

2.1 L'espai mètric ($\mathscr{H}(\boldsymbol{X})$,h)

Per començar a estudiar els fractals, cal determinar l'espai on estudiar-los i dotar-lo d'una mètrica. De base treballarem en espais mètrics $(\boldsymbol{X}, d)$ coneguts com $(\mathbb{R}^2, \text{Euclidiana})$. Però quan volem estudiar subconjunts "blanc-i-negre" de l'espai es torna natural introduir l'espai $\mathscr{H}$.

Definició 2.1. *Sigui $(\boldsymbol{X}$,d) un espai mètric complet. Denotem per $\mathscr{H}(\boldsymbol{X})$ l'espai que té per punts els subconjunts compactes no buits de $\boldsymbol{X}$. És a dir, $\mathscr{H}(\boldsymbol{X})$ és la col·lecció de tots els subconjunts compactes no buits de $\boldsymbol{X}$.*

Per obtenir un espai mètric, cal dotar aquest espai d'una mètrica que mesuri la distància entre els elements del conjunt. Cal tenir en compte la dualitat següent: sigui $A \in \mathscr{H}(\boldsymbol{X})$ ens podrem referir a A com un subconjunt de l'espai mètric $(\boldsymbol{X}, d)$ o bé com un punt o element de l'espai mètric $(\mathscr{H}(\boldsymbol{X}), h)$ que definirem a continuació.

Definició 2.2. *Sigui $(\boldsymbol{X}, d)$ un espai mètric complet. Definim la distància del punt $x \in \boldsymbol{X}$ al conjunt $B \in \mathscr{H}(\boldsymbol{X})$ com*

$$d(x, B) = \min\{d(x, y) : y \in B\}.$$

Ben definida: Cal demostrar que sempre podem trobar un mínim en el conjunt $\{d(x, y) : y \in B\}$. Veurem que és conseqüència de que $B \in \mathscr{H}(\boldsymbol{X})$ sigui un conjunt compacte i no buit. Considerem la funció $f : B \to \mathbb{R}$ definida com

$$f(y) = d(x, y) \quad \text{per a tot } x, y \in B.$$

Per la definició de mètrica se segueix que f és contínua en veure-la com a transformació de l'espai mètric (B, d) en l'espai mètric $(\mathbb{R}, \text{Euclidiana})$. Considerem $P = Inf\{f(y) : y \in B\}$, el qual està ben definit. Donat que $0 \leq f(y) < \infty$ per a tot $y \in B$, se segueix que P és finit. Volem veure que existeix un punt $\hat{y} \in B$ tal que $d(x, \hat{y}) = P$. Com podem trobar una successió infinita de punts $\{y_n : n = 1, 2, 3, \dots\} \subset B$ tal que $|f(y_n) - P| < 1/n$, per la compacitat de B podem trobar una subsuccessió d'aquesta tal que tingui per límit $\hat{y} \in B$. Per la continuïtat de f obtenim que $f(\hat{y}) = P$, com volíem veure.

Definició 2.3. *Sigui $(\boldsymbol{X}, d)$ un espai mètric complet. Definim la distància del conjunt $A \in \mathscr{H}(\boldsymbol{X})$ al conjunt $B \in \mathscr{H}(\boldsymbol{X})$ com*

$$d(A, B) = \max\{d(x, B) : x \in A\}.$$

Ben definida: De nou, utilitzant la compacitat de A i B, es pot demostrar que tal distància està ben definida. En particular, que existeixen punts $\hat{x} \in A$ i $\hat{y} \in B$ tal que $d(A, B) = d(\hat{x}, \hat{y})$.

Malgrat haver definit així la distància, observem que d no pot ser una mètrica en $\mathscr{H}(\boldsymbol{X})$ perquè no compleix la propietat simètrica: siguin $A, B \in \mathscr{H}(\mathbf{X})$, la distància de A a B no equival sempre a la distància de B a A. Aquest fet motiva la definició següent:

Definició 2.4. *Sigui $(\boldsymbol{X}, d)$ un espai mètric complet. Definim la distància de Hausdorff entre els punts $A, B \in \mathscr{H}(\boldsymbol{X})$ com*

$$h(A, B) = d(A, B) \vee d(B, A)$$

on utilitzem la notació $x \vee y$ per denotar el màxim de dos nombres reals x i y.

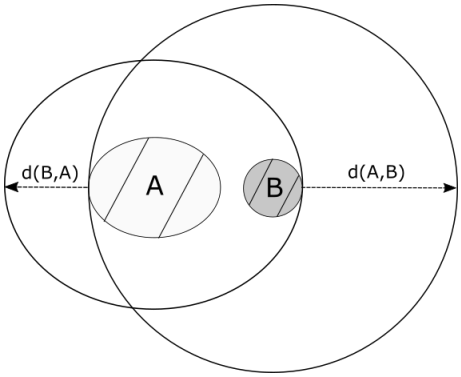

Figura 3: Distància de Hausdorff.

Per definició i el que havíem vist anteriorment, $h(A,B) = d(a,b)$ per a alguns $a \in A$ i $b \in B$. A més, aquesta distància sí que forma una mètrica com volíem.

Proposició 2.5. *Sigui $(\boldsymbol{X}, d)$ un espai mètric, aleshores $(\mathscr{H}(\boldsymbol{X}), h)$ és un espai mètric.*

Demostració: Cal demostrar que h és una mètrica en $\mathscr{H}(\boldsymbol{X})$.

(i) $h(A,B) = h(B,A) \quad \forall A, B \in \mathscr{H}(\boldsymbol{X})$: Siguin $A, B \in \mathscr{H}(\boldsymbol{X})$, $h(A,B) = d(A,B) \vee d(B,A) = d(B,A) \vee d(A,B) = h(B,A)$.

(ii) $0 < h(A,B) < \infty \quad \forall A, B \in \mathscr{H}(\boldsymbol{X})$, $A \neq B$: Siguin $A, B \in \mathscr{H}(\boldsymbol{X})$, tenim $h(A,B) = d(a,b)$ per a alguns $a \in A$ i $b \in B$ i aleshores $0 \leq h(A,B) < \infty$. Però, com $A \neq B$, existeix $a \in A$ tal que $a \notin B$ i tenim $h(A,B) \geq d(a,B) > 0$.

(iii) $h(A,A) = 0 \quad \forall A \in \mathscr{H}(\boldsymbol{X})$: Sigui $A \in \mathscr{H}(\boldsymbol{X})$, $h(A,A) = d(A,A) \vee d(A,A) = d(A,A) = \max\{d(x,A) : x \in A\} = \max\{\min\{d(x,y) : y \in A\} : x \in A\} = 0$.

(iv) $h(A,B) \leq h(A,C) + h(C,B) \quad \forall A, B, C \in \mathscr{H}(\boldsymbol{X})$: Siguin $A, B, C \in \mathscr{H}(\boldsymbol{X})$. Primer cal veure $d(A,B) \leq d(A,C) + d(C,B)$. Per a $a \in A$ qualsevol

$$\begin{aligned} d(a,B) &= \min\{d(a,b) : b \in B\} \\ &\leq \min\{d(a,c) + d(c,b) : b \in B\} \, \forall c \in C \\ &= d(a,c) + \min\{d(c,b) : b \in B\} \, \forall c \in C. \end{aligned}$$

Si es compleix per a tot punt $c \in C$ en concret es compleix per a un dels punts més propers a a, és a dir, es compleix per a $c' \in \{c \in C : d(a,c') = min\{d(a,c) : c \in C\}\}$. Així doncs

$$\begin{aligned} d(a,B) &\leq d(a,c') + \min\{d(c',b) : b \in B\} \\ &= \min\{d(a,c) : c \in C\} + \min\{d(c',b) : b \in B\} \\ &\leq \min\{d(a,c) : c \in C\} + \max\{\min\{d(c,b) : b \in B\} : c \in C\} \\ &= d(a,C) + d(C,B). \end{aligned}$$

Com és vàlid per a tot $a \in A$, tenim $d(A,B) \leq d(A,C) + d(C,B)$. Anàlogament obtenim $d(B,A) \leq d(B,C) + d(C,A)$. Llavors

$$\begin{aligned} h(A,B) = d(A,B) \vee d(B,A) &\leq (d(A,C) + d(C,B)) \vee (d(B,C) + d(C,A)) \\ &\leq d(A,C) \vee d(C,A) + d(B,C) \vee d(C,B) = h(A,C) + h(C,B). \end{aligned}$$

Per tant, $(\mathscr{H}(\mathbf{X}), h)$ és un espai mètric i l'anomenarem l'**espai dels fractals**. De moment, definirem un fractal com qualsevol subconjunt d'aquest espai mètric.

2.2 La completitud de l'espai mètric ($\mathscr{H}(\boldsymbol{X})$,h)

Si parem per un moment a pensar en la creació d'un fractal artificial com el triangle de Sierpinski, ens adonem que el que definim com a fractal és el límit d'una successió d'elements de l'espai $(\mathscr{H}(\boldsymbol{X})), h)$. D'aquí doncs sorgeix la necessitat de preguntar-nos si aquesta successió té límit i si aquest límit pertany o no al mateix espai. Així doncs, l'objectiu d'aquest subapartat és establir que l'espai de fractals $(\mathscr{H}(\boldsymbol{X})), h)$ és un espai mètric complet i caracteritzar les successions convergents en $\mathscr{H}(\boldsymbol{X})$. Calen dos lemes inicials per aconseguir-ho.

Lema 2.6. *Siguin $A, B \in \mathscr{H}(\boldsymbol{X})$ on $(\boldsymbol{X}, d)$ és un espai mètric i $\epsilon > 0$. Aleshores*

$$h(A, B) \leq \epsilon \Leftrightarrow A \subset B + \epsilon \text{ i } B \subset A + \epsilon$$

on considerem la dilatació d'un subconjunt $S \subset \boldsymbol{X}$ per la bola de radi ϵ com $S + \epsilon = \{y \in \boldsymbol{X} : d(x, y) \leq \epsilon \text{ per a } x \in S\}$.

Demostració: Demostrem primer $d(A, B) \leq \epsilon \Leftrightarrow A \subset B + \epsilon$. Suposem $d(A, B) \leq \epsilon$, aleshores $\max\{d(a, B) : a \in A\} \leq \epsilon$, que implica $d(a, B) \leq \epsilon$ per a tot $a \in A$. Aleshores per a tot $a \in A$ es compleix $a \in B + \epsilon$ i, per tant, $A \subset B + \epsilon$ com volíem demostrar. Per demostrar la implicació contrària, suposem $A \subset B + \epsilon$. Volem veure que $d(A, B) = \max\{d(a, B) : a \in A\} \leq \epsilon$. Sigui $a \in A$ qualsevol, per hipòtesi existeix $b \in B$ tal que $d(a, b) \leq \epsilon$ i aleshores $d(a, B) \leq \epsilon$ per a cada $a \in A$. Per tant, $d(A, B) \leq \epsilon$ com volíem demostrar. Un cop tenim aquesta doble implicació és fàcil veure que es compleix el lema per la definició de la mètrica de Hausdorff: $h(A, B) = d(A, B) \vee d(B, A)$.

Lema 2.7. ***(Lema de l'extensió).*** *Siguin $(\boldsymbol{X}, d)$ un espai mètric complet, $\{A_n : n = 1, 2, \ldots, \infty\}$ una successió de Cauchy de punts en $(\mathscr{H}(\boldsymbol{X}), h)$ i $\{n_j\}_{j=1}^{\infty}$ una successió d'enters tal que $0 < n_1 < n_2 < n_3 < \cdots$.*

Suposem que tenim una successió de Cauchy $\{x_{n_j} \in A_{n_j} : j = 1, 2, 3, \ldots\}$ en $(\boldsymbol{X}, d)$. Aleshores existeix una successió de Cauchy $\{\tilde{x}_n \in A_n : n = 1, 2, \ldots\}$ tal que $\tilde{x}_{n_j} = x_{n_j}$ per a tot $j = 1, 2, 3, \ldots$.

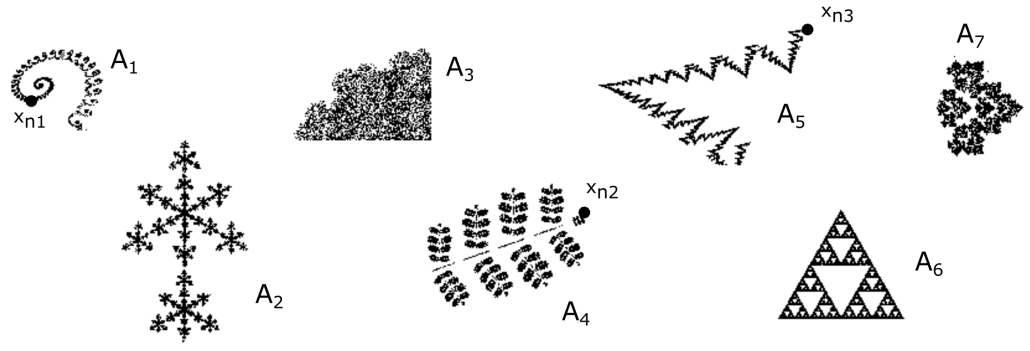

Figura 4: Principi d'una successió de Cauchy de conjunts $\{A_n\}$ de $(\mathscr{H}(\mathbf{X}), h)$ i subsuccessió de punts $\{x_{n_j}\}$ pertanyents a una subsuccessió dels conjunts.

Demostració: Per demostrar-ho, construirem primer una successió $\{\tilde{x}_n \in A_n : n = 1, 2, \ldots\}$ i seguidament veurem que compleix les condicions desitjades i que és de Cauchy.

Per a cada $n \in \{1, 2, \ldots, n_1\}$ escollim $\tilde{x}_n \in \{x \in A_n : d(x, x_{n_1}) = d(x_{n_1}, A_n)\}$, és a dir, $\tilde{x}_n$ és el punt més proper (o un dels més propers) de A_n al punt x_{n_1}. La existència d'aquest punt ve donada per la compacitat de A_n. De forma similar, per a cada $j \in \{1, 2, 3, \ldots\}$ i per a $n \in \{n_j + 1, \ldots, n_{j+1}\}$ escollim $\tilde{x}_n \in \{x \in A_n : d(x, x_{n_j}) = d(x_{n_j}, A_n)\}$.

Ara observem que la successió $\{\tilde{x}_n\}$ és de fet una extensió de $\{x_{n_j}\}$ en $\{A_n\}$, per a complir-se $\tilde{x}_{n_j} = x_{n_j}$ per a $j = 1, 2, 3, \ldots$ i $\tilde{x}_n \in A_n$ per a $n = 1, 2, \ldots$. Només falta veure que és de Cauchy, és a dir, per a $\epsilon > 0$ existeix $N > 0$ tal que per a $m, n \geq N$ es compleix $d(\tilde{x}_m, \tilde{x}_n) < \epsilon$. Considerem doncs $\epsilon > 0$. Per hipòtesi, com $\{x_{n_j}\}$ és una successió de Cauchy de punts en $(\boldsymbol{X}, d)$, tenim que existeix $N_1 > 0$ tal que per a $n_j, n_k > N_1$ es compleix $d(x_{n_j}, x_{n_k}) < \epsilon/3$. També per hipòtesi, com $\{A_n : n = 1, 2, \ldots, \infty\}$ és una successió de Cauchy de punts en $(\mathscr{H}(\boldsymbol{X}), h)$, tenim que existeix $N_2 > 0$ tal que per a $m, n > N_2$ es compleix $h(A_m, A_n) < \epsilon/3$ i, en conseqüència, $d(A_m, A_n) < \epsilon/3$. Considerant $N = \max\{N_1, N_2\}$, tenim

$$d(\tilde{x}_m, \tilde{x}_n) \leq d(\tilde{x}_m, x_{n_j}) + d(x_{n_j}, x_{n_k}) + d(x_{n_k}, \tilde{x}_n)$$

on $m \in \{n_{j-1} + 1, \ldots, n_j\}$ i $n \in \{n_{k-1}, \ldots, n_k\}$. Fixem-nos que, per com havíem escollit $\tilde{x}_m$, tenim $d(\tilde{x}_m, x_{n_j}) = d(x_{n_j}, A_m) \leq max\{d(x, A_m) : x \in A_{n_j}\} = d(A_{n_j}, A_m) < \epsilon/3$. Així doncs, $d(\tilde{x}_m, x_{n_j}) < \epsilon/3$ i de forma anàloga $d(x_{n_k}, \tilde{x}_n) < \epsilon/3$, obtenint $d(\tilde{x}_m, \tilde{x}_n) < \epsilon$ per a $m, n > N$ com volíem.

Teorema 2.8. ***(La completitud de l'espai de fractals).*** *Sigui $(\boldsymbol{X}, d)$ un espai mètric complet. Aleshores $(\mathscr{H}(\boldsymbol{X}), h)$ és un espai mètric complet. A més a més, si $\{A_n \in \mathscr{H}(\boldsymbol{X})\}_{n=1}^{\infty}$ és una successió de Cauchy aleshores*

$$A = \lim_{n \to \infty} A_n \in \mathscr{H}(\boldsymbol{X}).$$

Es pot caracteritzar A com

$$A = \{x \in \boldsymbol{X} : \textit{existeix una successió } \{x_n \in A_n\} \textit{ que convergeix a } x\}.$$

Demostració: Per demostrar que $(\mathscr{H}(\boldsymbol{X}), h)$ és un espai mètric complet ens falta veure que tota successió de Cauchy $\{A_n \in \mathscr{H}(\boldsymbol{X})\}_{n=1}^{\infty}$ té límit $A \in \mathscr{H}(\boldsymbol{X})$. Per fer-ho, considerem $\{A_n\}$ una successió de Cauchy en $\mathscr{H}(\boldsymbol{X})$ i A com s'ha definit al teorema. Partim la demostració en cinc parts, sent la tercera part essencial de les següents:

(i) $A \neq \emptyset$.

(ii) A tancat.

(iii) per a $\epsilon > 0$ exiteix $N > 0$ tal que per a $n \geq N$, $A \subset A_n + \epsilon$.

(iv) A totalment fitat.

(v) $Lim\, A_n = A$.

A partir de (ii) i (iv) s'obté que A és compacte i, com també és no buit per (i), obtenim que $A \in \mathscr{H}(\boldsymbol{X})$, part de la cosa que volíem demostrar.

Demostració de (i): Cal provar l'existència d'una successió de Cauchy $\{a_i \in A_i\}$ en $\boldsymbol{X}$ que convergeix a $a \in \boldsymbol{X}$. Pel fet que $\{A_n\}$ és una successió de Cauchy en $\mathscr{H}(\mathbf{X})$, podem trobar una successió d'enters positius $N_1 < N_2 < N_3 < \cdots < N_n < \cdots$ tal que

$$h(A_m, A_n) \leq \frac{1}{2^i} \quad \text{per a } m, n \geq N_i.$$

Escollim $x_{N_1} \in A_{N_1}$. Com $h(A_{N_1}, A_{N_2}) \leq \frac{1}{2}$ podem trobar un $x_{N_2} \in A_{N_2}$ tal que $d(x_{N_1}, x_{N_2}) \leq \frac{1}{2}$. Suposem haver escollit una successió finita $x_{N_i} \in A_{N_i}$; $i = 1, 2, \ldots, k$ per la qual $d(x_{N_{i-1}}, x_{N_i}) \leq 1/2^{i-1}$. Gràcies a que $h(A_{N_k}, A_{N_{k+1}}) \leq 1/2^k$ i $x_{N_k} \in A_{N_k}$ podem trobar $x_{N_{k+1}} \in A_{N_{k+1}}$ tal que $d(x_{N_k}, x_{N_{k+1}}) \leq 1/2^k$. Per inducció, podem trobar una successió infinita $x_{N_i} \in A_{n_i}$ tal que $d(x_{N_i}, x_{N_{i+1}}) \leq 1/2^i$. Per veure que $\{x_{N_i}\}$ és una successió de Cauchy en $\boldsymbol{X}$, considerem $\epsilon > 0$ i escollim N_ϵ tal que $\sum_{i=N_\epsilon}^{\infty} 1/2^i < \epsilon$. Aleshores per a $n > m \geq N_\epsilon$ tenim

$$d(x_{N_m}, x_{N_n}) \leq d(x_{N_m}, x_{N_{m+1}}) + \cdots + d(x_{N_{n-1}}, x_{N_n}) < \sum_{i=N_\epsilon}^{\infty} \frac{1}{2^i} < \epsilon.$$

Pel lema d'extensió, podem extendre la successió a una altra successió de Cauchy $\{a_i \in A_i\}$ tal que $a_{N_i} = x_{N_i}$. Aleshores existeix $Lim\, a_i$ i per definició pertany a A. Per tant, $A \neq \emptyset$.

Demostració de (ii): Cal demostrar que sigui $\{a_i \in A\}$ una successió que convergeix a un punt a, tal punt pertany a A. D'una banda, per hipòtesi existeix una succesió creixent d'enters positius $\{N_i\}_{i=1}^{\infty}$ tal que $d(a_{N_i}, a) < 1/i$. D'altra banda, per definició dels elements d'A, per a cada enter positiu i existeix una successió $\{x_{i,n} \in A_n\}$ tal que $\lim_{n\to\infty} x_{i,n} = a_i$. Gràcies a això podem trobar una succesió d'enters $\{m_i\}$ tal que $d(x_{N_i,m_i}, a_{N_i}) \leq 1/i$, és a dir, $d(x_{N_1,m_1}, a_{N_1}) \leq 1$, $d(x_{N_2,m_2}, a_{N_2}) \leq 1/2$, etc. Així doncs, tenim $d(x_{N_i,m_i}, a) < 2/i$. Considerant doncs $y_{m_i} = x_{N_i,m_i}$ veiem que $y_{m_i} \in A_{m_i}$ i $\lim_{i\to\infty} y_{m_i} = a$. Pel lema d'extensió, $\{y_{m_i}\}$ es pot extendre a una successió convergent $\{z_i \in A_i\}$ que tindrà també límit $a \in A$. Per tant, A és tancat.

Demostració de (iii): Cal demostrar que per a $\epsilon > 0$ existeix $N > 0$ tal que per a $n \geq N$, $A \subset A_n + \epsilon$. Considerem $\epsilon > 0$. Per ser $\{A_n\}$ successió de Cauchy, existeix $N > 0$ tal que per a $m, n \geq N$ es compleix $h(A_m, A_n) \leq \epsilon$. Fixant $n \geq N$, per a $m \geq n$ tenim doncs $A_m \subset A_n + \epsilon$. Volem veure que la propietat és vàlida al límit també, és a dir, $A \subset A_n + \epsilon$. Per fer-ho, considerem $a \in A$ i veurem que pertany a $A_n + \epsilon$. Tenim per definició d'A que existeix una successió $\{a_i \in A_i\}$ que convergeix a a. Suposem que la N és també suficientment gran en complir-se que $d(a_m, a) < \epsilon$ per a $m \geq N$. Aleshores $a_m \in A_n + \epsilon$ degut a que $A_m \subset A_n + \epsilon$. I ara com A_n és compacte, es pot demostrar que $A_n + \epsilon$ és tancat. Llavors com $a_m \in A_n + \epsilon$ per a tot $m \geq N$, a també pertany a $A_n + \epsilon$. Per tant, $A \subset A_n + \epsilon$ per a n suficientment gran.

Demostració de (iv): Suposem que A no és totalment fitat. Aleshores per alguna $\epsilon > 0$ no existeix una ϵ-xarxa finita. Podem trobar una successió $\{x_i\}_{i=1}^{\infty}$ en A tal que $d(x_i, x_j) \geq \epsilon$ per a $i \neq j$. Veurem que això arriba a contradicció. Per (iii) existeix una n suficientment gran tal que $A \subset A_n + \epsilon/3$. Per a cada x_i de la successió podem trobar una corresponent $y_i \in A_n$ per la qual $d(x_i, y_i) \leq \epsilon/3$. Ara bé, com A_n és compacte, alguna subsuccessió $\{y_{n_i}\}$ de $\{y_i\}$ convergeix i en conseqüència serà de Cauchy, pel que podem trobar punts de la successió $\{y_{n_i}\}$ tan a prop com volem. En particular, podem trobar dos punts y_{n_i} i y_{n_j} tal que $d(y_{n_i}, y_{n_j}) < \epsilon/3$. Però aleshores

$$d(x_{n_i}, x_{n_j}) \leq d(x_{n_i}, y_{n_i}) + d(y_{n_i}, y_{n_j}) + d(y_{n_j}, x_{n_j}) < \frac{\epsilon}{3} + \frac{\epsilon}{3} + \frac{\epsilon}{3}$$

i tenim una contradicció amb la manera en que s'havia definit $\{x_{n_i}\}$. Per tant, A és totalment fitat.

Demostració de (v): Pel raonament fet a la demostració principal tenim que $A \in \mathscr{H}(\boldsymbol{X})$. Cal veure $Lim\, A_n = A$, per definició, donat $\epsilon > 0$ existeix $N > 0$ tal que $\forall n > N$ es compleix $h(A_n, A) < \epsilon$, o equivalentment, $A_n \subset A + \epsilon$ i $A \subset A_n + \epsilon$. Donat que la segona

inclusió ja l'hem demostrat, només cal veure que sigui $\epsilon > 0$ existeix $N > 0$ tal que $\forall n > N$ es compleix $A_n \subset A + \epsilon$. Considerem $\epsilon > 0$ qualsevol. Pel fet que $\{A_n\}$ és una succesió de Cauchy, podem trobar $N > 0$ tal que per a $n, m \geq N$ $h(A_m, A_n) \leq \epsilon/2$. Aleshores per a $m, n \geq N$, $A_n \subset A_m + \epsilon/2$. Sigui $n \geq N$, de la mateixa manera podem trobar que existeix una succesió d'enters $\{N_i\}$ complint $n < N_1 < N_2 < \ldots < N_k < \ldots$ tal que per a $m, k \geq N_j$ es té $A_m \subset A_k + \epsilon/2^{j+1}$. Fixem-nos que així obtenim $A_n \subset A_{N_1} + \epsilon/2$, $A_{N_1} \subset A_{N_2} + \epsilon/2^2$, ... Sigui $y \in A_n$, existeix $x_{N_1} \in A_{N_1}$ tal que $d(y, x_{N_1}) \leq \epsilon/2$. També existeix $x_{N_2} \in A_{N_2}$ tal que $d(x_{N_1}, x_{N_2}) \leq \epsilon/2^2$. De forma similar per inducció podem trobar una successió $x_{N_1}, x_{N_2}, x_{N_3}, \ldots$ tal que $x_{N_j} \in A_{n_j}$ i $d(x_{N_j}, x_{N_{j+1}}) < \epsilon/2^{j+1}$. Així obtenim que $\{x_{N_j}\}$ és una successió de Cauchy i, en conseqüència, que té límit $x \in A$. Utilitzant la desigualtat triangular uns quants cops podem veure

$$d(y, x_{N_j}) \leq \frac{\epsilon}{2} + \frac{\epsilon}{2^2} + \ldots + \frac{\epsilon}{2^j} \leq \epsilon \sum_{i=1}^{\infty} \frac{1}{2^i} \leq \frac{\epsilon}{3} \leq \epsilon \quad \text{per a tot } j.$$

Com $d(y, x_{N_j}) \leq \epsilon$, també $d(y, x) \leq \epsilon$ i obtenim així que $A_n \subset A + \epsilon$ per a $n \geq N$. Això completa la demostració que $Lim\, A_n = A$ i conseqüentment que $(\mathscr{H}(\boldsymbol{X}), h)$ és un espai mètric complet.

Tot plegat, hem demostrat que tota successió de Cauchy en l'espai dels fractals $(\mathscr{H}(\boldsymbol{X}), h)$ convergeix i el límit es troba també en aquest espai. Això ens permet dir que allò que entenem de moment com a fractal, el límit d'una successió d'elements de l'espai $(\mathscr{H}(\boldsymbol{X})), h)$, pertany també a l'espai de fractals.

2.3 Transformacions

En aquest punt, l'objectiu és conèixer diferents tipus de transformacions que ens permetin crear les successions d'elements de l'espai de fractals $(\mathscr{H}(\boldsymbol{X}), h)$. Per fer-ho, definirem primer les transformacions en l'espai base $(\boldsymbol{X}, d)$ i poc a poc estendrem els resultats a l'espai de fractals.

Definició 2.9. *Sigui $(\boldsymbol{X}, d)$ un espai mètric. Una transformació en $\boldsymbol{X}$ és una funció $f : \boldsymbol{X} \to \boldsymbol{X}$ que assigna un únic punt $f(x) \in \boldsymbol{X}$ a cada punt $x \in \boldsymbol{X}$. Si $S \subset \boldsymbol{X}$ aleshores $f(S) = \{f(x) : x \in S\}$. Direm que:*

- *f és injectiva si per a $x, y \in \boldsymbol{X}$ complint $f(x) = f(y)$ aleshores $x = y$.*
- *f és exhaustiva si $f(\boldsymbol{X}) = \boldsymbol{X}$.*
- *f és bijectiva o invertible si és injectiva i exhaustiva alhora.*

Si f és invertible és possible definir la transformació inversa $f^{-1} : \boldsymbol{X} \to \boldsymbol{X}$ per $f^{-1}(y) = x$ on $x \in \boldsymbol{X}$ és l'únic punt complint $y = f(x)$.

Sigui $f : \mathbf{X} \to \mathbf{X}$ una transformació en un espai mètric. Les iteracions endavant de f són les transformacions $f^{\circ n} : \mathbf{X} \to \mathbf{X}$ definides per $f^{\circ 0}(x) = x, f^{\circ 1}(x) = f(x), f^{\circ 2}(x) = f(f(x)), \ldots, f^{\circ(n+1)}(x) = f \circ (f^n(x))$ per a $n = 0, 1, , 2, \ldots$. Si f és invertible, es poden definir les iteracions enrere de f com les transformacions $f^{\circ(-m)}(x) : \mathbf{X} \to \mathbf{X}$ definides com $f^{\circ(-1)}(x) = f^{-1}(x), f^{\circ(-m)}(x) = (f^{\circ m})^{-1}(x)$ per a $m = 1, 2, 3, \ldots$.

De transformacions en podem trobar de molt diferents segons l'espai en el qual s'estigui treballant. Alguns exemples podrien ser: transformacions polinomials en la recta real,

transformacions afins en el pla euclidià o les transformacions de Möbius en l'esfera de Riemann. Cal remarcar que, en la geometria fractal determinista, es posa el focus en aquells subconjunts "complicats" de l'espai generats per transformacions "senzilles" de l'espai en sí mateix. Com centrarem l'interés en l'espai $(\mathbb{R}^2, \text{Euclidiana})$, presentem les transformacions següents:

Definició 2.10. *Una transformació $w : \mathbb{R}^2 \to \mathbb{R}^2$ de la forma*

$$w(x_1, x_2) = (ax_1 + bx_2 + e, cx_1 + dx_2 + f)$$

on a, b, c, d, e i f són nombres reals, s'anomena una transformació afí dos-dimensional. Sovint utilitzem la notació equivalent

$$w(x) = w\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix} = Ax + t.$$

Així doncs, la transformació afí general $w(x) = Ax+t$ en $\mathbb{R}^2$ consisteix en una transformació lineal, A, que deforma l'espai en relació amb l'origen que es manté fix, seguida d'una translació especificada pel vector t. La matriu A es pot escriure sempre en coordenades polars de la forma

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} r_1 \cos\theta_1 & -r_2 \sin\theta_2 \\ r_1 \sin\theta_1 & r_2 \cos\theta_2 \end{pmatrix}$$

on (r_1, θ_1) són les coordenades polars del punt (a, c) i $(r_2, \theta_2 + \pi/2)$ les del punt (b, d).

La combinació de diferents transformacions lineals bàsiques com la rotació d'angle θ R_θ, l'escalat S_v o la reflexió R (amb les matrius següents) o altres transformacions lineals juntament amb les translacions donen lloc a moltes transformacions afins.

$$R_\theta = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \qquad S_v = \begin{pmatrix} v_x & 0 \\ 0 & v_y \end{pmatrix} \qquad R = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

2.4 Teorema de l'aplicació contractiva

La varietat de transformacions que es poden dur a terme en un espai donat és molt gran. No obstant, en la geometria fractal ens interessa un tipus de transformacions concret, les transformacions contractives, aquelles que apropen els punts de l'espai en cada iteració.

Definició 2.11. *Una transformació $f : \boldsymbol{X} \to \boldsymbol{X}$ en un espai mètric $(\boldsymbol{X}, d)$ s'anomena contractiva o aplicació contractiva si existeix una constant $0 \leq s < 1$ tal que*

$$d(f(x), f(y)) \leq s\, d(x, y)\, \forall x, y \in \boldsymbol{X}.$$

Tal nombre s s'anomena factor de contracció de f.

Fixem-nos que pel moment ens restringim a l'espai base. Aquestes transformacions són d'especial interès pel teorema següent, també conegut com a *Teorema del punt fix de Banach*, que garanteix l'existència i unicitat de punts fixos de certes funcions en espais mètrics i dóna un mètode per trobar-los.

Teorema 2.12. ***(Teorema de l'aplicació contractiva)*** *Sigui $f : \boldsymbol{X} \to \boldsymbol{X}$ una aplicació contractiva en un espai mètric complet $(\boldsymbol{X}, d)$. Aleshores f posseeix exactament un punt fix $x_f \in \boldsymbol{X}$ i a més a més per a qualsevol punt $x \in \boldsymbol{X}$, la successió $\{f^{\circ n}(x) : n = 0, 1, 2, \dots\}$ convergeix a x_f. És a dir,*

$$\lim_{n\to\infty} f^{\circ n}(x) = x_f \quad \textit{per a cada } x \in \boldsymbol{X}.$$

Demostració: Dividim la demostració en tres parts:

1. La successió $\{f^{\circ n}\}$ té límit en $\boldsymbol{X}$. Si veiem que $\{f^{\circ n}\}$ és una successió de Cauchy, en ser $\boldsymbol{X}$ complet, tindrem que la successió té límit $x_f \in \boldsymbol{X}$. Sigui $x \in \boldsymbol{X}$. Siguin $x \in \boldsymbol{X}$ i $0 \leq s < 1$ un factor de contracció de f. Fàcilment podem veure que es compleix
$$d(f^{\circ n}(x), f^{\circ m}(x)) \leq s^{m \wedge n} d(x, f^{\circ |n-m|})(x)$$
per a tot $m, n = 0, 1, 2, \ldots$ per a x fixada. La notació $u \wedge v$ denota el mínim del parell de nombres reals u i v. També podem veure que
$$\begin{aligned} d(x, f^{\circ n}(x)) &\leq d(x, f(x)) + d(f(x), f^{\circ 2}(x)) + \cdots + d(f^{\circ (k-1)}(x), f^{\circ k}(x)) \\ &\leq (1 + s + s^2 + \cdots + s^{k-1})\, d(x, f(x)) \\ &\leq (1-s)^{-1} d(x, f(x)) \end{aligned}$$
aprofitant la suma de la sèrie geomètrica. Tot plegat obtenim
$$d(f^{\circ n}(x), f^{\circ m}(x)) \leq s^{m \wedge n} (1-s)^{-1} d(x, f(x))$$
de la qual cosa es dedueix que $f^{\circ n}(x)_{n=0}^{\infty}$ és una successió de Cauchy ja que com més iteracions es fan més petita és la distància. Per tant, com $\boldsymbol{X}$ és complet, la successió té límit $x_f \in \boldsymbol{X}$, és a dir,
$$\lim_{n \to \infty} f^{\circ n}(x) = x_f.$$

2. x_f és un punt fix de f. Tenim f és contractiva i aleshores contínua, per tant:
$$f(x_f) = f\left(\lim_{n \to \infty} f^{\circ n}(x)\right) = \lim_{n \to \infty} f^{\circ (n+1)}(x) = x_f.$$

3. x_f és l'únic punt fix. Suposem que hi ha dos punts fixos x_f i y_f de f. Aleshores $x_f = f(x_f)$ i $y_f = f(y_f)$ i tenim
$$d(x_f, y_f) = d(f(x_f), f(y_f)) \leq s\, d(x_f, y_f).$$
Però llavors $(1-s)\, d(x_f, y_f) \leq 0$ que implica $d(x_f, y_f) = 0$ i per tant $x_f = y_f$. Això finalitza la demostració.

2.5 Aplicacions contractives a l'espai de fractals

L'objectiu d'aquest subapartat és estendre aquests últims resultats, referents a les aplicacions contractives i al teorema de l'aplicació contractiva, de l'espai base $(\boldsymbol{X}, d)$ a l'espai de fractals $(\mathscr{H}(\boldsymbol{X}), h)$. Primerament, cal caracteritzar les aplicacions contractives en $(\mathscr{H}(\boldsymbol{X}), h)$ i ho farem amb els tres lemes següents.

Lema 2.13. *Sigui $w : \boldsymbol{X} \to \boldsymbol{X}$ una aplicació contractiva en l'espai mètric $(\boldsymbol{X}, d)$. Aleshores w és contínua.*

Demostració: Sigui $\epsilon > 0$ donat. Considerant s el factor de contracció de w, aleshores
$$d(w(x), w(y)) \leq s\, d(x, y) < \epsilon$$
sempre que $d(x, y) < \delta$ on $\delta = \epsilon / s$. Això completa la demostració.

Lema 2.14. *Sigui $w : \boldsymbol{X} \to \boldsymbol{X}$ una aplicació contractiva en l'espai mètric $(\boldsymbol{X}, d)$. Aleshores w envia $\mathscr{H}(\boldsymbol{X})$ en sí mateix.*

Demostració: Sigui S un subconjunt no buit de $\boldsymbol{X}$. Clarament $w(S) = \{w(x) : x \in S\}$ és no buit. Volem veure que $w(S)$ és també compacte. Sigui $\{y_n = w(x_n)\}$ una successió de punts en S. Aleshores $\{x_n\}$ és una successió infinita de punts en S. Com S és compacte, existeix una subsuccesió $\{x_{N_n}\}$ que convergeix a un punt $\hat{x} \in S$. Però aleshores la continuitat de w implica que $\{y_{N_n} = w(x_{N_n})\}$ és una subsuccessió de $\{y_n\}$ que convergeix a $\hat{y} = f(\hat{x}) \in w(S)$.

Lema 2.15. *Sigui $w : \boldsymbol{X} \to \boldsymbol{X}$ una aplicació contractiva en un espai mètric $(\boldsymbol{X}, d)$ amb factor de contracció s. Aleshores $w : \mathscr{H}(\boldsymbol{X}) \to \mathscr{H}(\boldsymbol{X})$ definida com*

$$w(B) = \{w(x) : x \in B\} \quad \forall B \in \mathscr{H}(\boldsymbol{X})$$

és una aplicació contractiva en $(\mathscr{H}(\boldsymbol{X}), h)$ amb factor de contracció s.

Demostració: Pels dos lemes anteriors tenim que $w : \boldsymbol{X} \to \boldsymbol{X}$ és contínua i també que w envia $\mathscr{H}(\boldsymbol{X})$ en sí mateix. Per com s'ha definit w, es té que $w : \mathscr{H}(\boldsymbol{X}) \to \mathscr{H}(\boldsymbol{X})$ assigna un únic punt $w(B) \in \mathscr{H}(\boldsymbol{X})$ a cada punt $B \in \mathscr{H}(\boldsymbol{X})$ i per tant és una transformació en $(\mathscr{H}(\boldsymbol{X}), h)$. Falta veure que és contractiva en aquest espai mètric amb factor de contracció s. Siguin $B, C \in \mathscr{H}(\boldsymbol{X})$, aleshores

$$\begin{aligned} d(w(B), w(C)) &= \max\{\min\{d(w(x), w(y)) : y \in C\} : x \in B\} \\ &\leq \max\{\min\{s\, d(x, y) : y \in C\} : x \in B\} = s\, d(B, C). \end{aligned}$$

De forma similar $d(w(C), w(B)) \leq s\, d(C, B)$ Així doncs

$$\begin{aligned} h(w(B), w(C)) &= d(w(B), w(C)) \vee d(w(C), w(B)) \leq s\, d(B, C) \vee d(C, B) \\ &= s\, h(B, C) \end{aligned}$$

obtenint $h(w(B), w(C)) \leq s\, h(B, C)$ com volíem demostrar.

El lema següent ens dóna un mètode important per combinar aplicacions contractives en $(\mathscr{H}(\mathbf{X}), h)$ per tal de crear noves aplicacions contractives del mateix espai mètric. Aquest mètode és diferent del típic mètode de composició.

Lema 2.16. *Siguin $(\boldsymbol{X}, d)$ un espai mètric, $\{w_n : n = 1, 2, \ldots, N\}$ aplicacions contractives en $(\mathscr{H}(\boldsymbol{X}), h)$ i s_n el factor de contracció de w_n per a tot n. Definim $W : \mathscr{H}(\boldsymbol{X}) \to \mathscr{H}(\boldsymbol{X})$ per*

$$W(B) = w_1(B) \cup w_2(B) \cup \cdots w_N(B) = \bigcup_{n=1}^{N} w_n(B) \quad \textit{per a } B \in \mathscr{H}(\boldsymbol{X}).$$

Aleshores W és una aplicació contractiva amb factor de contracció $s = max\{s_n : n = 1, 2, \ldots, N\}$.

Demostració: Tenim que W és una transformació per definir-se a partir de les diferents w_n, que són transformacions. Cal demostrar doncs la contractivitat de W. Demostrarem que és cert per a $N = 2$ i utilitzarem que per a tot $B, C, D, E \in \mathscr{H}(\mathbf{X})$ es compleix $h(B \cup C, D \cup E) \leq h(B, D) \vee h(C, E)$ on h és la mètrica de Hausdorff. Siguin $B, C \in \mathscr{H}(\boldsymbol{X})$, observem que es compleix

$$\begin{aligned} h(W(B)W(C)) &= h(w_1(B) \cup w_2(B), w_1(C) \cup w_2(C)) \\ &\leq h(w_1(B), w_1(C)) \vee h(w_2(B), w_2(C)) \\ &\leq s_1\, h(B, C) \vee s_2\, h(B, C) \leq s\, h(B, C). \end{aligned}$$

Un argument inductiu completaria la demostració.

Ara sí, tot plegat ens dirigeix a definir la construcció matemàtica següent i arribar al resultat buscat.

Definició 2.17. *Un sistema iteratiu de funcions consisteix en un espai mètric complet* $(\boldsymbol{X}, d)$ *juntament amb un conjunt finit d'aplicacions contractives* $w_n : \boldsymbol{X} \to \boldsymbol{X}$ *amb respectius factors de contracció* s_n*, per a* $n = 1, 2, \ldots, N$*. L'abreviació IFS ve donada per l'anglès "Iterated Function System". La notació per l'IFS anunciat és* $\{\boldsymbol{X}; w_n, n = 1, 2, \ldots, N\}$ *i el seu factor de contracció és* $s = max\{s_n : n = 1, 2, \ldots, N\}$*.*

Al llarg de la memòria denotem per afinitats cadascuna de les aplicacions contractives que formen un IFS.

Teorema 2.18. *Sigui* $\{\boldsymbol{X}; w_n, n = 1, 2, \ldots, N\}$ *un sistema iteratiu de funcions amb factor de contracció* s*. Aleshores la transformació* $W : \mathscr{H}(\boldsymbol{X}) \to \mathscr{H}(\boldsymbol{X})$ *definida com*

$$W(B) = \bigcup_{n=1}^{N} w_n(B) \quad \textit{per a tot } B \in \mathscr{H}(\boldsymbol{X})$$

és una aplicació contractiva en l'espai mètric complet $(\mathscr{H}(\boldsymbol{X}), h)$ *amb factor de contracció* s*. És a dir,*

$$h(W(B), W(C)) \leq s\, h(B, C) \quad \textit{per a tot } B, C \in \mathscr{H}(\boldsymbol{X}).$$

Pel teorema de l'aplicació contractiva, el seu únic punt fix, $A \in \mathscr{H}(\boldsymbol{X})$*, compleix*

$$A = W(A) = \bigcup_{n=1}^{N} w_n(A)$$

i ve donat per $A = \lim_{n\to\infty} W^{\circ n}(B)$ *per a tot* $B \in \mathscr{H}(\boldsymbol{X})$*.*

Demostració: La demostració ve donada clarament pels resultats previs al teorema.

Definició 2.19. *El punt fix* $A \in \mathscr{H}(\boldsymbol{X})$ *descrit al teorema anterior s'anomena atractor de l'IFS.*

I ara sí, podem definir un **fractal determinista** com l'atractor d'un IFS, és a dir, el punt fix d'una aplicació contractiva en $(\mathscr{H}(\boldsymbol{X}), h)$ sota certes condicions respecte l'espai $(\boldsymbol{X}, d)$ i les aplicacions contractives implicades.

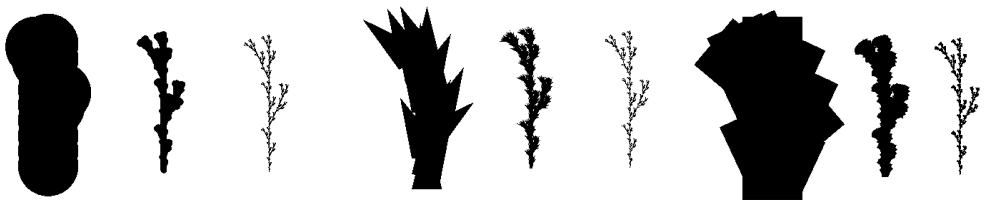

Figura 5: Diferents iteracions de l'algoritme determinista per un mateix IFS i diferents elements inicials.

La unicitat de l'atractor d'un IFS es mostra a la Figura 5 on s'utilitza un dels algoritmes descrits al proper apartat per crear fractals donat un IFS. Es pot veure, tant sols observant algunes de les primeres iteracions, com el punt fix de l'IFS no depèn del punt o element de l'espai de fractals escollit inicialment, en aquests casos un cercle, un triangle i un quadrat.

2.6 Dos algoritmes per calcular fractals a partir d'IFS

Un cop definits els fractals, dedicarem la nostra atenció a trobar la manera de generar-los. Ens centrarem en IFS de la forma $\{\mathbb{R}^2, w_i : i = 1, 2, \ldots, N\}$, pel que les seves afinitats seran de la forma que havíem vist en les transformacions,

$$w_i(x) = w_i \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix},$$

i per tant s'expressen en taules com la següent, concretament aquesta representa l'IFS de l'anterior subapartat que té cinc afinitats:

$\boldsymbol{w}$	$\boldsymbol{a}$	$\boldsymbol{b}$	$\boldsymbol{c}$	$\boldsymbol{d}$	$\boldsymbol{e}$	$\boldsymbol{f}$
1	0.33	0	0	0.33	0	0
2	0.33	0	0	0.33	0	0.33
3	0.33	0	0	0.33	0	0.66
4	0.29	0.13	-0.13	0.29	0	0.33
5	0.29	-0.13	0.13	0.29	0	0.66

Vet aquí dos algoritmes per representar imatges dels atractors d'IFS en un ordinador.

1. **Algoritme determinista**: utilitzant el resultat del teorema anterior, es basa en la idea de calcular directament una successió de conjunts $\{A_n : W^{\circ n}(A)\}$ començant per un conjunt inicial A_0.

   Sigui $\{\boldsymbol{X}; w_1, w_2, \ldots, w_N\}$ un IFS. Escollim un conjunt compacte $A_0 \subset \boldsymbol{X}$. Aleshores calculem successivament $A_n = W^{\circ n}(A)$ mitjançant

   $$A_{n+1} = \bigcup_{j=1}^{N} w_j(A_n) \quad \text{per a } n = 1, 2, \ldots$$

   Això construeix una successió $\{A_n : n = 0, 1, 2, 3, \ldots\} \subset \mathscr{H}(\boldsymbol{X})$. Pel teorema anterior la successió $\{A_n\}$ convergeix a l'atractor de l'IFS en la mètrica de Hausdorff.

   Seguidament un exemple de les primeres iteracions d'aquest algoritme. En aquest cas, a simple vista ràpidament el resultat s'aproxima a l'atractor.

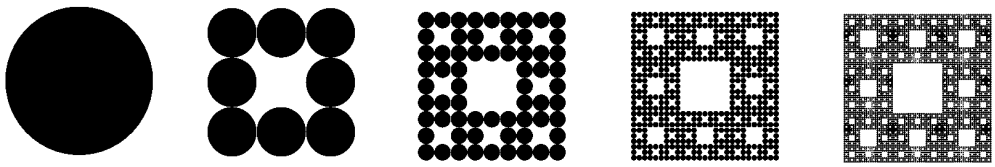

Figura 6: Exemple de les primeres iteracions de l'algoritme determinista.

2. **Algoritme d'iteració aleatòria**: es basa en la idea de calcular directament una successió de punts $\{x_n\}_{n=0}^{\infty}$ començant per un punt inicial x_0 i escollint cada cop una aplicació w_n per a $n = 1, 2, \ldots, N$ aleatòriament amb certes probabilitats.

   Sigui $\{\boldsymbol{X}; w_1, w_2, \ldots, w_N\}$ un IFS, on s'assigna una probabilitat $p_i > 0$ a cada w_i per a $i = 1, 2, \ldots, N$ amb $\sum_{i=1}^{N} p_i = 1$. Escollim $x_0 \in \boldsymbol{X}$. Aleshores calculem recursivament, de forma independent,

   $$x_n \in \{w_1(x_{n-1}, w_2(x_{n-1}), \ldots, w_N(x_{n-1})\} \quad \text{per a } n = 1, 2, 3, \ldots$$

on la probabilitat de $x_n = w_i(x_{n-1})$ és p_i. Això construeix una successió $\{x_n : n = 0, 1, 2, 3, \dots\} \subset \boldsymbol{X}$. La successió $\{x_n\}$ "convergeix" a l'atractor de l'IFS sota diverses condicions que s'estudien en detall en la teoría ergòdica (dedicada a l'estudi del comportament mitjà a llarg termini dels sistemes dinàmics). Tals probabilitats juguen un paper molt important a l'hora de dibuixar fractals amb aquest algoritme.

Aprofitant que el factor d'escala amb què una transformació afí canvia d'àrea ve donat pel determinant de la seva part lineal, les probabilitats es poden prendre aproximadament:

$$p_i \approx \frac{|\det A_i|}{\sum_{j=1}^{N} |\det A_j|} = \frac{|a_i d_i - b_i c_i|}{\sum_{j=1}^{N} |a_j d_j - b_j c_j|} \quad \text{on } i = 1, 2, \dots, N. \tag{2.1}$$

En el cas que algun dels determinants sigui nul, cal assignar un nombre positiu petit com 0.001.

A continuació un exemple que mostra els diferents resultats que s'obtenen en utilitzar l'algoritme d'iteració aleatòria en un mateix IFS en augmentar el nombre d'iteracions.

Figura 7: Exemple de diferents nombres d'iteracions de l'algoritme determinista.

2.7 Conjunts de condensació

No obstant, aquests dos algoritmes no ens permeten crear l'arbre fractal que es va presentar a la introducció. Si pensem en aquest com part d'un IFS, podem pensar que aquest està format per dues transformacions que redueixen la mida de l'arbre, el traslladen a la part superior del tronc i el roten cadascuna amb un angle diferent. No obstant, ens faltaria definir d'alguna manera el tronc... Per fer-ho, presentarem en aquesta secció una altra forma de crear aplicacions contractives en $(\mathscr{H}(\boldsymbol{X}), h)$ que ens serà útil per crear arbres fractals i donarem els corresponents resultats, que seran anàlegs als anteriors.

Definició 2.20. *Siguin $(\boldsymbol{X}, d)$ un espai mètric i $C \in \mathscr{H}(\boldsymbol{X})$. Definim una transformació $w_0 : \mathscr{H}(\boldsymbol{X}) \to \mathscr{H}(\boldsymbol{X})$ per $w_0(B) = C$ per a tot $B \in \mathscr{H}(\boldsymbol{X})$. Aleshores w_0 s'anomena transformació de condensació i C s'anomena el conjunt de condensació associat a w_0.*

Observem que la transformació de condensació $w_0 : \mathscr{H}(\boldsymbol{X}) \to \mathscr{H}(\boldsymbol{X})$ és una aplicació contractiva en l'espai mètric $(\mathscr{H}(\boldsymbol{X}), h)$ amb factor de contracció igual a zero i posseix un únic punt, el mateix conjunt de condensació. En afegir una transformació de condensació a un IFS obtenim un IFS amb condensació, que compleix clarament el Teorema 2.18 anàleg. Per crear l'arbre que volíem, només caldrà afegir una tercera transformació que sigui de condensació al tronc inicial.

Definició 2.21. *Siguin* $\{\boldsymbol{X}; w_1, w_2, \ldots, w_N\}$ *un IFS amb factor de contracció* $0 \leq s < 1$ *i* $w_0 : \mathscr{H}(\boldsymbol{X}) \to \mathscr{H}(\boldsymbol{X})$ *una transformació de condensació. Aleshores* $\{\boldsymbol{X}; w_0, w_1, \ldots, w_N\}$ *s'anomena sistema iteratiu de funcions (IFS) amb condensació, amb factor de contracció* s*.*

Teorema 2.22. *Sigui* $\{\boldsymbol{X}; w_n, n = 0, 1, 2, \ldots, N\}$ *un sistema iteratiu de funcions amb condensació, amb factor de contracció* s*. Aleshores la transformació* $W : \mathscr{H}(\boldsymbol{X}) \to \mathscr{H}(\boldsymbol{X})$ *definida com*

$$W(B) = \bigcup_{n=0}^{N} w_n(B) \quad \textit{per a tot } B \in \mathscr{H}(\boldsymbol{X})$$

és una aplicació contractiva en l'espai mètric complet $(\mathscr{H}(\boldsymbol{X}), h)$ *amb factor de contracció* s*. És a dir,*

$$h(W(B), W(C)) \leq s\, h(B, C) \quad \textit{per a tot } B, C \in \mathscr{H}(\boldsymbol{X}).$$

Pel teorema de l'aplicació contractiva, el seu únic punt fix, $A \in \mathscr{H}(\boldsymbol{X})$*, compleix*

$$A = W(A) = \bigcup_{n=0}^{N} w_n(A)$$

i ve donat per $A = \lim_{n\to\infty} W^{\circ n}(B)$ *per a tot* $B \in \mathscr{H}(\boldsymbol{X})$*.*

Aprofitant aquest nou mètode, ara sí que podem generar arbres fractals dos-dimensionals de molts tipus. A la Figura 8 es poden observar algunes mostres que exemplifiquen la varietat de formes i figures que es poden crear amb aquest tipus d'IFS en variar els paràmetres com el nombre de branques, la mida d'aquestes, els angles, etc.

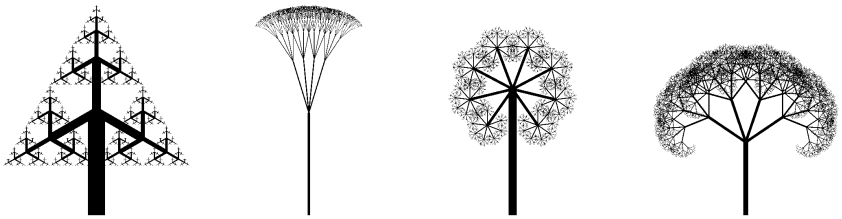

Figura 8: Arbres fractals dos-dimensionals.

2.8 Fractals amb l'ajuda del Teorema del Collage

En aquest punt sabem com obtenir l'atractor d'un IFS donat. El resultat següent ens permetrà dur a terme el procés invers: obtenir un IFS amb atractor proper a un conjunt donat.

Teorema 2.23. *Sigui* $(\boldsymbol{X}, d)$ *un espai mètric complet,* $L \in \mathscr{H}(\boldsymbol{X})$ *i* $\epsilon \geq 0$ *donat. Escollim un IFS (o IFS amb condensació)* $\{\boldsymbol{X}; (w_o), w_1, w_2, \ldots, w_N\}$ *amb factor de contracció* $0 \leq s < 1$ *tal que*

$$h\left(L, \bigcup_{\substack{n=1 \\ (n=0)}}^{N} w_n(L)\right) \leq \epsilon$$

on h és la mètrica de Hausdorff. Aleshores

$$h(L, A) \leq \epsilon/(1-s)$$

on A és l'atractor de l'IFS. Equivalentment

$$h(L, A) \leq (1-s)^{-1} h\left(L, \bigcup_{\substack{n=1 \\ (n=0)}}^{N} w_n(L)\right) \quad \textit{per a tot } L \in \mathscr{H}(\boldsymbol{X}).$$

Demostració: Ve donada pel lema següent.

Lema 2.24. *Siguin ($\boldsymbol{X}$,d) un espai mètric complet, $f : \boldsymbol{X} \to \boldsymbol{X}$ una aplicació contractiva amb factor de contracció $0 \leq s < 1$ i $x_f \in \boldsymbol{X}$ el punt fix de f. Aleshores*

$$d(x, x_f) \leq (1-s)^{-1} d(x, f(x)) \quad \textit{per a tot } x \in \boldsymbol{X}.$$

Demostració: La funció distància $d(a, b)$ per a $a \in \boldsymbol{X}$ fixada és contínua en $b \in \boldsymbol{X}$. Aleshores

$$\begin{aligned} d(x, x_f) &= d\left(x, \lim_{n\to\infty} f^{\circ n}(x)\right) = \lim_{n\to\infty} d(x, f^{\circ n}(x)) \\ &\leq \lim_{n\to\infty} \sum_{m=1}^{n} d(f^{\circ(m-1)}(x), f^{\circ(m)}(x)) \\ &\leq \lim_{n\to\infty} d(x, f(x))(1 + s + \ldots + s^{n-1}) \leq (1-s)^{-1} d(x, f(x)). \end{aligned}$$

El teorema doncs ens diu que per trobar un IFS que tingui un atractor "proper" o "semblant" al donat, cal trobar un conjunt de transformacions (aplicacions contractives en un espai adequat dins el qual visqui el conjunt donat) tal que la unió o collage de les imatges del conjunt donat sota aquestes transformacions sigui propera al conjunt donat. La proximitat es mesura amb la mètrica de Hausdorff. I quina millor manera de veure el funcionament que amb un exemple.

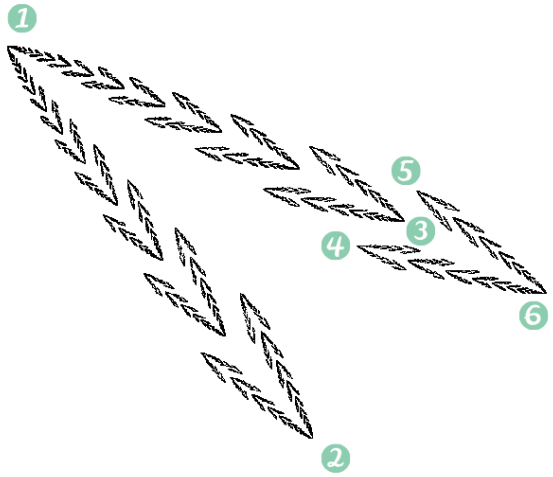

Figura 9: Fractal i alguns dels punts més característics que el formen.

Suposem que tenim el fractal bidimensional de la Figura 9 però no coneixem l'IFS que el genera. Segons el Teorema del Collage, cal doncs trobar les transformacions que formen l'IFS i, en fixar-s'hi bé, es pot observar que aquest IFS ve donat per dues transformacions. Recordem que, com ens trobem en l'espai dos-dimensional, les transformacions són de la forma

$$w_i \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix} \quad \text{on } i \in \{1, 2\}$$

pel que per cadascuna de les transformacions es tenen sis incògnites: a, b, c, d, e i f. La manera de trobar aquestes incògnites és dotant la figura d'unes coordenades, trobar tres punts i les seves respectives imatges per cadascuna de les transformacions i resoldre cadascun dels sistemes lineals determinats amb sis equacions i sis incògnites. Així doncs, dotant la figura d'un sistema de coordenades centrat en la part inferior esquerra, obtenim la taula següent:

Punt	(x_1, x_2)	Imatge per w_1	$w_1(x_1, x_2)$	Imatge per w_2	$w_2(x_1, x_2)$
(1)	(0.02,5,19)	(1)	(0.02,5.19)	(6)	(6.00,2.48)
(2)	(3.34,0.81)	(3)	(4.38,3.31)	(5)	(4.58,3.61)
(6)	(6.00,2.48)	(2)	(3.34,0.81)	(4)	(3.92,2.97)

Amb aquests valors, resolent els dos sistemes d'equacions resultants, podem obtenir els valors que determinen l'IFS generador del fractal que buscàvem.

$\boldsymbol{w}$	$\boldsymbol{a}$	$\boldsymbol{b}$	$\boldsymbol{c}$	$\boldsymbol{d}$	$\boldsymbol{e}$	$\boldsymbol{f}$
1	0.16	-0.88	-0.82	-0.19	4.56	6.20
2	-0.31	0.09	-0.05	-0.30	5.53	4.03

2.9 La dependència contínua dels fractals en els paràmetres

Fins al moment només hem centrat l'atenció en el fractals deterministes, aquells que no inclouen aleatorietat. En aquest subapartat ens preguntem què passa en afegir paràmetres en un IFS i com afecta això a l'atractor d'aquest. Per estudiar-ho, el primer pas és considerar que introduïm un paràmetre en una de les aplicacions que constitueixen l'IFS, obtenint el següent.

Lema 2.25. *Siguin (P, d_p) i $(\boldsymbol{X}, D)$ espais mètrics on el segon també és complet. Sigui $w : P \times \boldsymbol{X} \to \boldsymbol{X}$ una família d'aplicacions contractives en $\boldsymbol{X}$ amb factor de contracció $0 \leq s < 1$, és a dir, per a cada $p \in P$, $w(p, \cdot)$ és una aplicació contractiva en $\boldsymbol{X}$. Sigui w contínua en P per a cada punt $x \in \boldsymbol{X}$. Aleshores el punt fix de w depèn contínuament en p o, anàlogament, la funció $x_f : P \to \boldsymbol{X}$ és contínua.*

Demostració: Considerem $x_f(p)$ el punt fix de w per a un paràmetre $p \in P$ fix. Sigui $p \in P$, aleshores per a tot $p \in P$

$$\begin{aligned} d(x_f(p), x_f(q)) &= d(w(p, x_f(p)), w(q, x_f(q))) \\ &\leq d(w(p, x_f(p)), w(q, x_f(p))) + d(w(q, x_f(p)), w(q, x_f(q))) \\ &\leq d(w(p, x_f(p)), w(q, x_f(p))) + s\, d(x_f(p), x_f(q)) \end{aligned}$$

que implica doncs

$$d(x_f(p), x_f(q)) \leq (1 - s)^{-1} d(w(p, x_f(p)), w(q, x_f(p)))$$

però el terme de la dreta es pot fer tan arbitràriament petit com calgui restringint q a ser suficientment propera a p com calgui. Això demostra la continuïtat de la funció x_f.

Ara cal veure que aquesta continuïtat es manté per a l'aplicació contractiva W que es defineix en l'IFS.

Lema 2.26. *Sigui $(\boldsymbol{X}, d)$ un espai mètric i suposem tenir transformacions continues $w_n : \boldsymbol{X} \to \boldsymbol{X} \quad (n = 1, 2, \dots N)$ depenent contínuament en el paràmetre $p \in P$, on (P, d_p) és un espai mètric compacte, és a dir, $w_n(p, x)$ depèn contínuament en p per a $x \in \boldsymbol{X}$ fixada. Aleshores la transformació $W : \mathscr{H}(\boldsymbol{X}) \to \mathscr{H}(\boldsymbol{X})$ definida per*

$$W(p, B) = \bigcup_{n=0}^{N} w_n(p, B) \quad \textit{per a tot } B \in \mathscr{H}(\boldsymbol{X})$$

és també contínua en p, és a dir, $W(p, B)$ és contínua en p a cada $B \in \mathscr{H}(\boldsymbol{X})$ en l'espai mètric $(\mathscr{H}(\boldsymbol{X}), h)$.

Demostració: És suficient demostrar pel cas $N = 1$ perquè el resultat es podrà estendre fàcilment amb la propietat de la mètrica de Hausdorff h següent: per a tot $B, C, D, E \in \mathscr{H}(\mathbf{X})$ es compleix $h(B \cup C, D \cup E) \leq h(B, D) \vee h(C, E)$. Sigui $B \in \mathscr{H}(\mathbf{X})$ volem veure que per a $p, q \in P$ i $\epsilon > 0$ donats existeix δ tal que

$$h(w_1(p, B), w_1(q, B)) < \epsilon \quad \text{si } d_p(p, q) < \delta.$$

Per definició de la mètrica de Hausdorff, cal veure que $d(w_1(p, B), w_1(q, B)) < \epsilon$ així com $d(w_1(q, B), w_1(p, B)) < \epsilon$ si $d(p, q) < \delta$. Com les demostracions són anàlogues, veurem només el primer cas:

$$\begin{aligned} d(w_1(p, B), w_1(q, B)) = &\max_{x \in B} \min_{y \in B} d(w_1(p, x), w_1(q, y)) \\ &\max_{x \in B} \min_{y \in B} \{d(w_1(p, x), w_1(p, y)) + d(w_1(p, y), w_1(q, y))\}. \end{aligned}$$

Ara $P \times B$ és compacte i $w_1 : P \times B \to \boldsymbol{X}$ és contínua, per tant w_1 és uniformement contínua i es compleix que existeix $\delta > 0$ tal que $d(w_1(p, y), w_1(q, y)) < \epsilon$ per a tot $y \in B$ quan $d_p(p, q) < \delta$. Per tant, assumint que $d_p(p, q) < \delta$ tenim

$$\begin{aligned} d(w_1(p, B), w_1(q, B)) < &\max_{x \in B} \min_{y \in B} \{d(w_1(p, x), w_1(p, y)) + \epsilon\} \\ &d(w_1(p, B), w_1(p, B)) + \epsilon = \epsilon \end{aligned}$$

com volíem demostrar.

La continuïtat es manté i, per tant, combinant el dos lemes anteriors obtenim el resultat següent:

Teorema 2.27. *Siguin $(\boldsymbol{X}, d)$ un espai mètric i $\{\boldsymbol{X}, (w_0), w_1, \dots, w_N\}$ un IFS (amb condensació) amb factor de contractivitat s. Per a $n = 1, 2, \dots, N$, considerem w_n depenents contínuament del paràmetre $p \in P$, on P és un espai mètric compacte. Aleshores l'atractor $A(p) \in \mathscr{H}(\boldsymbol{X})$ depèn contínuament en $p \in P$ respecte la mètrica de Hausdorff h.*

En altres paraules, petits canvis en els paràmetres defineixen petits canvis en l'atractor sempre que el sistema es mantingui hiperbòlic. Per tant, podem controlar de forma

contínua l'atractor d'un IFS ajustant els paràmetres en les transformacions. Aquests resultats es poden veure en l'atractor d'IFS de la Figura 10 que varia de forma contínua en variar els paràmetres.

I fins aquí l'estudi teòric dels fractals que ens servirà de base per als següents apartats, dedicats a la generació de diferents tipus de fractals amb mètodes variats.

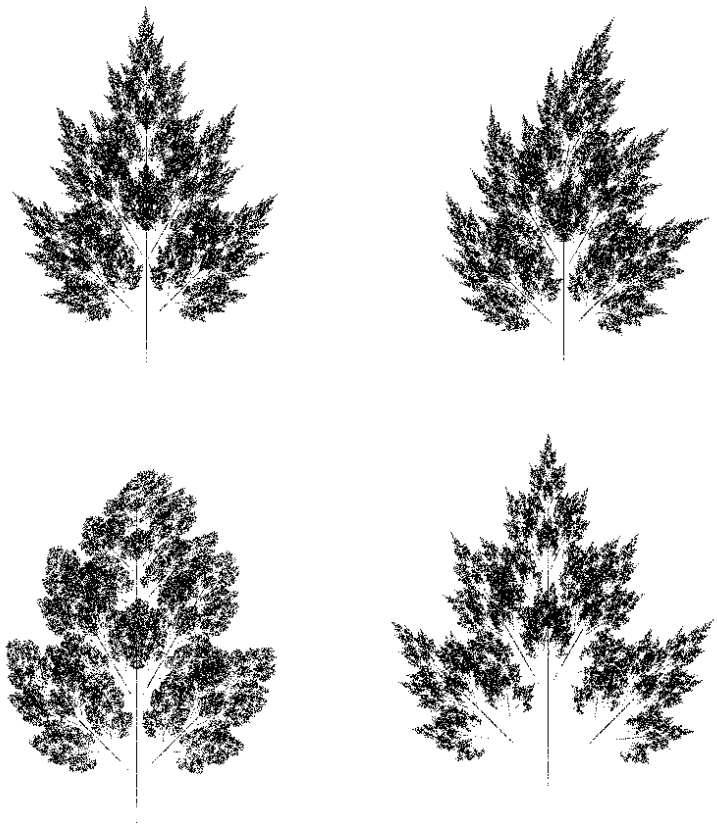

Figura 10: Resultats d'un mateix IFS sota la modificació de paràmetres.

3 El programari

El programari que s'utilitza per dur a terme els diferents programes és l'entorn integrat de desenvolupament *Visual Studio*, una eina informàtica per al desenvolupament de programari de manera còmoda i ràpida que s'adapta correctament a les nostres necessitats per admetre la implementació de diferents biblioteques o conjunts de funcions que són de gran utilitat per fer simulacions. El llenguatge utilitzat és *C++*. S'utilitzen dues biblioteques principals molt relacionades per generar els quatre programes.

D'una banda, la biblioteca *FLTK* permet la creació d'una interfície gràfica d'usuari on es poden visualitzar els resultats. També permet crear diferents controls des dels quals es poden modificar els paràmetres o l'escena com sigui convenient. Les funcions i estructures d'aquesta biblioteca comencen per *Fl_* com *Fl_Window* per iniciar la interfície o *Fl_Counter* per crear un control de comptatge.

D'altra banda, la biblioteca *OpenGL* és capaç de dibuixar escenes bidimensionals i tridimensionals complexes a partir de primitives geomètriques bàsiques. Les funcions d'aquesta biblioteca comencen per *gl* o *glut*.

Abans de centrar-nos en els diferents programes es considera necessari, pel seu rerefons matemàtic i la importància dins el programa creat, explicar com es dibuixen objectes amb primitives, quin procés es duu a terme internament per aconseguir mostrar el resultat en pantalla i quina és l'estructura bàsica comuna dels diferents programes.

3.1 El dibuix d'objectes amb primitives

Comencem pel dibuix d'objectes amb primitives. Per tal de dibuixar objectes, *OpenGL* defineix dues operacions bàsiques que es poden realitzar en qualsevol moment: dibuixar algun element com les primitives geomètriques (punts, línies, o polígons) i canviar l'estat o la forma amb què es dibuixen aquests elements.

- Les operacions de dibuixar primitives permeten crear totes les figures geomètriques especificant els seus vèrtexs. El tipus de primitiva determina com es combinaran els vèrtexs per formar la superfíce poligonal final. La creació de primitives es realitza entre *glBegin(PRIMITIVA)* i *glEnd()*, essent *PRIMITIVA* alguna de les primitives bàsiques com *GL_POINTS* per dibuixar punts o *GL_QUAD_STRIP* per crear una sèrie de quadrats adjacents. La implementació d'altres mòduls permet implementar superfícies més complexes com cilindres o esferes de forma automàtica, no obstant, també es poden definir amb més paciència amb les primitives bàsiques.
- Les operacions de canvi d'estat s'encarreguen d'inicialitzar les variables internes d'*OpenGL* que defineixen com es dibuixaran les primitives. Són molt variades i algunes de les formes més utilitzades són:
  - Gestió de vèrtexs: *glColor()* permet establir el color amb què es dibuixaran els vèrtexs, *glNormal()* per especificar la normal que s'utilitzarà per a la il·luminació, *glTexCoord()* per indicar coordenades de textura.
  - Activació de modes: mitjançant *glEnable()* i *glDisable()* es poden activar o desactivar característiques internes d'*OpenGL*.
  - Característiques especials: molt variades, com les referents a la il·luminació o els materials. Per la seva importància, es pot trobar un breu resum als annexos.

3.2 Pipeline de *OpenGL*

Continuem per entendre el procés que es duu a terme internament per mostrar el resultat en pantalla. En els diferents programes es treballa en l'espai tres-dimensional tot i que, en aquells en què els resultats són purament dos-dimensionals, s'anul·la la tercera component quedant tots els objectes sobre el pla $\{(x, y, z) \in \mathbb{R}^3 : z = 0\}$. La dificultat rau doncs en mostrar una visualització dos-dimensional per pantalla d'una escena tres-dimensional.

El procés de visualitzar una escena en 3D mitjançant gràfics a l'ordinador és similar a la que es realitza quan es fa un fotografia. En primer lloc, cal situar els objectes del món que es volen fotografiar, per exemple un bol ple de fruites diferents sobre una taula de fusta. A continuació, cal situar el trípode amb la càmara en un lloc determinat de l'espai i encarar-la allà on volem fotografiar. Finalment, quan disparem la fotografia, només una petita part del món ens queda representat a la imatge 2D final, la resta del món queda retallat i no apareix a la imatge.

Definim el pipeline gràfic com el conjunt de processos que es duen a terme internament a l'ordinador per part del programa des de la creació de l'escena 3D fins a la visualització 2D en pantalla. En aquest procés, els elements de l'escena pateixen diferents transformacions i varien en diferents coordenades. Les principals etapes són les següents:

- Transformacions de modelatge.
- Transformacions de visualització.
- Transformacions de projecció.
- Transformacions de retallada i pantalla.

Òbviament aquest procés és més senzill quan tots els objectes es dibuixen sobre un mateix pla, no obstant, el procés intern del programa és el mateix.

3.2.1 Les matrius 4x4

Totes aquestes transformacions es duen a terme mitjançant les transformacions afins que havíem vist a l'apartat anterior, aquest cop tres-dimensionals. Recordem però que la translació requereix realitzar una suma mentre la rotació i l'escalat requereixen multiplicacions. Per homogeneïtzar aquestes transformacions i realitzar-les de forma eficient (molt important per la repetida utilització d'aquestes) recorrem a la representació homogènia.

Definició 3.1. *Donat un punt (x, y, z) en l'espai $\mathbb{R}^3$, definim la seva representació homogènia com (x_h, y_h, z_h, h) on $h \neq 0$ i es compleix $x = x_h/h$, $y = y_h/h$ i $z = z_h/h$. Tal h s'anomena paràmetre homogeni.*

Observació 3.2. Existeixen infinites representacions homogènies equivalents per a cada punt, tot i que normalment s'utilitza $h = 1$ quedant la representació homogènia $(x, y, z, 1)$. Pels vectors de la forma (x, y, z) en l'espai $\mathbb{R}^3$ considerem la seva representació homogènia com $(x, y, z, 0)$.

Fixem-nos ara que, donat un punt $P \in \mathbb{R}^3$, podem obtenir el punt resultant per una translació, un escalat o una rotació (en aquest cas la rotació respecte l'eix X) anomenat

$P' \in \mathbb{R}^3$ de la forma $P' = M \times P$ on M és una de les matrius següents:

$$T = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad S = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad R_x = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.1}$$

D'aquesta manera, la composició o concatenació de transformacions correspon matemàticament a la simple multiplicació de matrius. Considerem que volem realitzar unes transformacions amb matrius $M_1, \ldots, M_n$ a un objecte poligonal format per un conjunt de vèrtexs o punts. És possible doncs obtenir l'anomenada matriu de transformació neta $M_{neta} = M_n \times \cdots \times M_1$ resultant de realitzar les successives transformacions als punts en l'ordre desitjat (recordem que la multiplicació de matrius no és commutativa). D'aquesta forma, només caldrà calcular un cop M_{neta} i multiplicar-la a cada punt del model per obtenir la seva posició final. Considerant P un d'aquests punts originals i P' el punt resultant tenim

$$P' = M_n \times \cdots \times M_1 \times P \quad \equiv \quad P' = M_{neta} \times P$$

3.2.2 Les piles de matrius

Fins aquí hem vist que *OpenGL* utilitza matrius 4×4 per representar totes les seves transformacions geomètriques. Internament treballa amb piles de matrius, de forma que únicament la matriu de la part superior de la pila és la que s'està utilitzant en un moment determinat. Hi ha tres piles de matrius principals, entre les quals és possible anar canviant utilitzant la funció *glMatrixMode()*.

1. Pila Modelview (*GL_MODELVIEW*). Aquesta pila conté les transformacions de modelatge i de visualització.
2. Pila Projection (*GL_PROJECTION*). Aquesta conté les matrius de projecció.
3. Pila Textura (*GL_TEXTURE*). Aquestes matrius s'utilitzen per transformar les coordenades de textura.

L'ús de piles de matrius ens facilita la construcció de models jeràrquics on és possible utilitzar models simples i unir-los per construir models més complexes, principalment pel fet de permetre tornar al sistema de coordenades anterior sense fer cap càlcul. Com treballem únicament amb la matriu de la part superior de la pila, ens serà d'utilitat la funció *glPushMatrix()*, que afegeix una còpia de la matriu de treball actual a la part superior de la pila, i la funció *glPopMatrix()* que elimina la matriu superior de la pila, descartant la seva informació i quedant-nos amb la matriu següent de la pila com a matriu activa.

3.2.3 Transformacions de modelatge

Les transformacions de modelatge són aquelles amb què els objectes es posicionen, modifiquen i orienten a l'escena. Originalment cada objecte té les seves coordenades del model a nivell local. Un cop es volen posicionar i orientar a nivell global passem a treballar amb les coordenades universals o del món.

Hi ha tres funcions bàsiques de modelatge que implementa *OpenGl* i que són de gran utilitat: *glTranslate(x,y,z)* per fer translacions, *glRotate(α,x,y,z)* per fer rotacions d'angle α respecte un eix donat i *glScale(x,y,z)* per fer escalats. Les matrius de la translació i l'escalat corresponen a les matrius T i S de l'equació 3.1, mentre que la rotació entorn a un eix arbitrari s'obte de la *fórmula de rotació de Rodrigues* i té la forma següent:

$$\begin{pmatrix} x^2(1-\cos\alpha)+\cos\alpha & xy(1-\cos\alpha)-z\sin\alpha & xz(1-\cos\alpha)+y\sin\alpha & 0 \\ yx(1-\cos\alpha)+z\sin\alpha & y^2(1-\cos\alpha)+\cos\alpha & yz(1-\cos\alpha)-x\sin\alpha & 0 \\ xz(1-\cos\alpha)-y\sin\alpha & yz(1-\cos\alpha)+x\sin\alpha & z^2(1-\cos\alpha)+\cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Per a cada objecte doncs, s'obté una matriu de modelatge final M_{model} resultat de combinar les diferents transformacions.

3.2.4 Transformacions de visualització

Les transformacions de visualització són aquelles amb què s'especifica la posició de la càmara i aquesta passa a ser l'origen del nou sistema de referència, apuntant en el sentit negatiu de l'eix Z i l'eix Y cap amunt, transformant en conseqüencia tota l'escena. Això permet passar de les coordenades universals o del món a les coordenades de visualització.

Per abans de definir aquesta transformació, *OpenGL* compta amb la funció *glLoadIdentity()* que carrega la matriu identitat com a matriu actual. Un cop fet això podem utilitzar la funció *gluLookAt()* per posicionar la càmara virtual i donar-li una orientació. Aquesta rep el punt *eye* (e_x, e_y, e_z) on se situa la càmara, el punt *center* (c_x, c_y, c_z) cap a on s'encara la càmara i el vector *up* (u_x, u_y, u_z) que marca la direcció vertical d'aquesta. Si no es crida aquesta funció, la càmara se situarà automàticament en l'origen de coordenades apuntant en el sentit negatiu de l'eix Z i l'eix Y cap amunt.

Com ho fa internament aquesta funció per modificar d'aquesta manera les coordenades? Primerament cal procedir a traslladar l'escena sencera per tal de quedar la càmara a l'origen de coordenades del món mitjançant la matriu de translació inversa a la posició de la càmara:

$$M_T = \begin{pmatrix} 1 & 0 & 0 & -e_x \\ 0 & 1 & 0 & -e_y \\ 0 & 0 & 1 & -e_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

A continuació cal fer una rotació adient per encarar la càmara en el sentit negatiu de l'eix Z i l'eix Y cap amunt. Per fer-ho, es calculen primer els vectors normalitzats *front*, *left* i *up** de l'objectiu a la càmara: $f = \frac{v}{||v||}$ on $v = (e_x - c_x, e_y - c_y, e_z - c_z)$, $l = \frac{w}{||w||}$ on $w = up \times f$ i $u = f \times l = \frac{up}{||up||}$.

Fer una rotació amb aquests vectors com a columnes transformaria els eixos cartesians als eixos que determinen els tres vectors, però volem tot just el contrari: que els eixos determinats pels tres vectors passin a ser els eixos cartesians. Per aquesta raó, només caldrà fer la rotació inversa i com aquesta matriu és ortogonal ($M \cdot M^T = Id \Rightarrow M^{-1} = M^T$) obtenim

$$M_R = \begin{pmatrix} l_x & u_x & f_x & 0 \\ l_y & u_y & f_y & 0 \\ l_z & u_z & f_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}^{-1} = \begin{pmatrix} l_x & u_x & f_x & 0 \\ l_y & u_y & f_y & 0 \\ l_z & u_z & f_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}^{T} = \begin{pmatrix} l_x & l_y & l_z & 0 \\ u_x & u_y & u_y & 0 \\ f_x & f_y & f_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Finalment, obtenim la matriu de visualització resultant com

$$M_{view} = M_R M_T = \begin{pmatrix} l_x & l_y & l_z & -l_x x_c - l_y y_c - l_z z_c \\ u_x & u_y & u_y & -u_x x_c - u_y y_c - u_z z_c \\ f_x & f_y & f_z & -f_x x_c - f_y y_c - f_z z_c \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

3.2.5 Transformacions de projecció

Les transformacions de projecció són aquelles amb què es converteix el volum de visualització (tots els elements que seran visualitzats) en un cub unitari de coordenades normalitzades.

Hi ha diferents tipus de projeccions, en el nostre cas utilitzarem la projecció ortogràfica (o paral·lela), la qual no modifica la mida dels objectes sense importar on estiguin situats respecte a la càmara. Considerant *dreta, esquerra, superior, inferior, lluny, aprop* els paràmetres que defineixen el volum de visualització a l'espai, la matriu de la projecció ortogonal, que es crida amb la funció *glOrtho()*, és la següent:

$$M_{\text{proj}} = \begin{pmatrix} \frac{2}{dreta-esquerra} & 0 & 0 & -\frac{dreta+esquerra}{dreta-esquerra} \\ 0 & \frac{2}{superior-inferior} & 0 & -\frac{superior+inferior}{superior-inferior} \\ 0 & 0 & -\frac{2}{lluny-aprop} & -\frac{lluny+aprop}{lluny-aprop} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Fixem-nos que correspon a la composició d'un escalat que redueix el volum de visualització al cub unitari amb una translació adient en el cas que alguna de les parelles de mides (com *dreta,esquerra*) no siguin la mateixa en valor absolut. El canvi de signe en la variable Z ve donat pel fet que es considera la part positiva aquella que està cap a dins de la càmara.

3.2.6 Transformacions de retallada i pantalla

Amb tot plegat, s'obté la matriu $M_{\text{MVP}} = M_{\text{proj}} \cdot M_{\text{view}} \cdot M_{\text{model}}$ per transformar l'espai tres-dimensional en el cub unitari. Per últim s'apliquen les transformacions de retallada i pantalla, aquelles amb les quals es retallen aquells objectes que es troben de forma parcial dins el cub unitari i, per a tot allò que queda dins el cub, s'obtenen les coordenades 2D de la finestra en pantalla.

3.3 L'estructura base

Finalment, abans de centrar-se en els diferents programes, ens fixarem en l'estructura base que tenen per entendre'ls millor. Tots els programes tenen una estructura base molt similar, dividida en quatre blocs principals. Els dos primers són comuns a tots els programes i els dos últims s'utilitzen o no segons la necessitat.

- **Control**: conté la funció principal o *main* i les funcions relatives a l'interfície gràfica i els controls. Des de la funció *main* es criden les funcions següents:
  - *ViewInit()*: estableix els paràmetres inicials amb *ParametersInit()*.
  - *ControlsInit()*: inicialitza els controls, els posiciona a l'interfície, estableix característiques com els valors màxims i mínims entre d'altres i marca les accions que es duen a terme en utilitzar-los amb les funcions *name_counter_cb()*.
  - *Reshape()*: remodela la interfície en modificar la mida d'aquesta.
  - *Display()*: inicia la pila Projection establint el tipus de projecció ortogonal i la pila Modelview cridant la funció *DisplayObjects()* per dibuixar els objectes amb les funcions del bloc *View*. En algun cas, també es crida *OutputsDisplay()* per mostrar certs valors en la interfície.
  - *Mouse()*: determina les accions a fer en clicar el ratolí.
  - *Motion() i PassiveMotion()*: estableixen les conseqüencies de moure tant la càmara o punt de vista com només el ratolí.
  - *Keyboard()* i *SpecialInput()*: controlen les accions per teclat.
  - *Idle()*: s'activa quan no es fa res en l'interfície i determina què fer.
  - *Read()*: Llegeix les dades de fitxer amb les funcions del bloc *InOut*.
  - *Compute()*: Fa els càlculs necessaris per dibuixar els objectes amb les funcions del bloc *Compute*.
- **View**: engloba les funcions on es dibuixen els objectes, formant la part essencial de cada programa. Aquestes funcions es descriuran a continuació als detalls d'implementació dels diferents programes corresponen a aquest bloc.
- **InOut**: comprèn les funcions de lectura d'arxius, ja siguin els valors dels IFS com de les textures, amb les funcions *IFSA_Read()* i *Textures_Load()* respectivament.
- **Compute**: inclou les funcions de càlcul, concretament de valors dels IFS.

Aquests blocs es combinen amb un fitxer *.h* que els uneix internament. I ara sí, un cop vistes totes les eines disponibles i les funcions principals dels programes, ens centrarem en els diferents programes duts a terme.

3.4 Els programes

Centrant-nos en el propòsit dels diferents programes, concretament se n'han creat quatre. Els dos primers corresponen als algoritmes per calcular fractals artificials a partir de l'IFS que s'han vist a l'apartat anterior, un duu a terme l'algoritme determinista i l'altre l'algoritme aleatori. Els altres dos se centren en generar un tipus concret de fractals, els arbres fractals o *fractal trees* en anglès. En un programa es creen en dues dimensions a partir dels IFS amb condensació i l'algoritme determinista i en l'altre es creen en tres dimensions amb una generació recursiva i visualització 3D. Els resultats d'aquest últim programa simulen fractals naturals d'una forma molt realista.

4 IFS Algoritme Determinista

4.1 Objectiu

L'objectiu d'aquest programa és crear fractals dos-dimensionals a partir de l'algoritme determinista. Recordem que aquest es basa en el càlcul d'una successió de punts o elements de l'espai de fractals a partir d'un punt inicial iterat successivament per la transformació generada per les aplicacions de l'IFS.

4.2 Introducció

En el nostre cas, considerem com a punt inicial de l'espai de fractals una figura geomètrica dos-dimensional senzilla situada a l'espai tres-dimensional (com s'ha comentat abans, amb la tercera component nul·la), les aplicacions de l'IFS són transformacions tres-dimensionals donades en forma de matrius 4x4 i es calcula la successió fins a un cert nivell que determina el paràmetre *Level* per evitar la saturació del programa.

Amb les eines que ofereix *OpenGL*, és difícil utilitzar l'algoritme determinista directament com diu la definició, ja que a partir de la figura geomètrica inicial caldria generar diferents còpies per aplicar a cadascuna una aplicació de l'IFS, procés que no es pot fer amb aquesta eina. No obstant, podem fer-ho d'una manera alternativa i equivalent. Començant per la identitat, fem totes les combinacions possibles fins a nivell *Level* de les afinitats de l'IFS corresponent i en acabar cadascuna dibuixem la figura corresponent en el sistema de coordenades del model que hagi quedat. El resultat final serà el mateix, tantes figures com determini el nombre d'afinitats de l'IFS elevat a *Level* que generaran l'atractor de l'IFS en augmentar suficientment el nivell.

4.3 Detalls d'implementació

El primer que cal fer és llegir de fitxer les dades dels IFS, obtenint les aplicacions que els defineixen en un conjunt de matrius 4x4. Un cop fet això, cal definir també les funcions que, a partir d'uns paràmetres *Radius, Height* i *Reduction*, dibuixen les figures geomètriques gràcies als vèrtexs que les defineixen, en aquest cas *drawEquilateral()*, *drawTriangle()* i *drawCercle()*. Llavors es pot definir la funció recursiva *DisplayRIFS(level)* a la qual inicialment es dóna el valor del paràmetre *Level* i, començant per la matriu identitat, s'encarrega de calcular totes les combinacions d'aplicacions possibles de l'IFS fins aquest nivell dibuixant al final de cada cop la figura geomètrica corresponent. L'estructura és la següent:

1. Si $level == 0$ dibuixem la figura geomètrica.
2. Per a tantes afinitats com tingui l'IFS: copiem la matriu, la multipliquem per la afinitat seleccionada, cridem *DisplayRIFS(level-1)* i en acabar esborrem la matriu.

D'aquesta recursivitat és important fixar-se en la utilitat que tenen les piles de matrius, que eviten calcular totes les combinacions d'afinitats possibles des de zero utilitzant cada cop la combinació d'afinitats calculada abans i un cert ordre. Això evita una gran quantitat de càlculs i simplifica molt el programa.

4.4 Utilitats

Per poder entendre de forma visual com treballen les afinitats sobre la figura inicial s'inclouen dues utilitats. D'una banda, es pinten de diferents colors els resultats d'aplicar les diferents aplicacions de l'IFS en el nivell que marca el paràmetre *LEVEL*. Això es fa situant al bucle dins el punt 2 de la recursivitat un canvi de color en arribar al nivell *LEVEL*. De l'altra, amb el botó "Levels" podem decidir dibuixar totes les diferents figures que resulten de variar el nivell de recursivitat entre 1 i *Level*. Això es decideix a la funció *IFSA_Display()* que es crida des de *DisplayObjects()*, i que decidirà de quina manera es crida la recursivitat. En cas de dibuixar les diferents figures, es pot modificar la seva posició a la interfície amb els controls anomenats *horizontal* i *vertical*. Per últim, també s'ofereix l'opció de mostrar els resultats en blanc i negre per centrar l'atenció només en la figura creada.

4.5 Resultats

Tot plegat, i deixant de banda el funcionament del programa, aquest ens permet veure com els resultats relatius a l'algoritme determinista i a l'atractor de l'IFS es compleixen: l'atractor de l'IFS no varia en modificar el punt inicial de l'espai de fractals que s'escull, sempre i quan el nivell sigui suficientment gran. A més, en la majoria d'IFS ràpidament es comença a intuir l'atractor, no cal un nivell massa alt de recursivitat.

Vet aquí un exemple del funcionament del programa que exemplifica molt bé el que s'ha comentat:

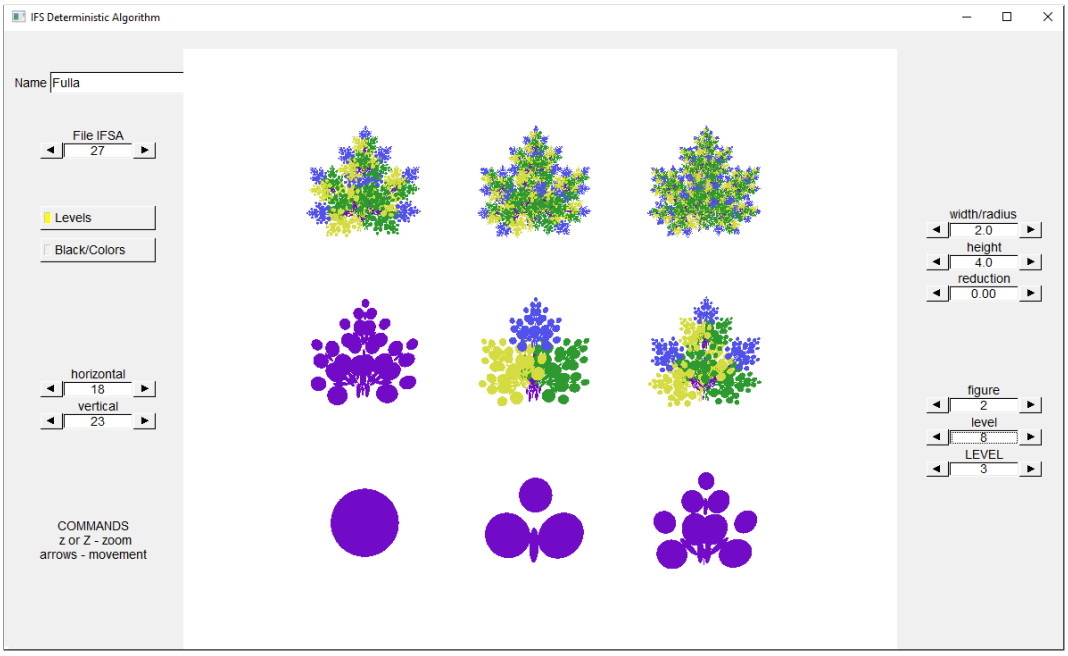

Figura 11: Interfície gràfica del programa *IFS Algoritme Determinista*.

5 IFS Algoritme Aleatori

5.1 Objectiu

L'objectiu d'aquest programa és crear fractals dos-dimensionals a partir de l'algoritme aleatori. Recordem que aquest es basa en calcular directament una successió de punts de l'espai, començant per un punt inicial i iterant-lo escollint cada cop una aplicació de l'IFS de forma aleatòria amb certes probabilitats.

5.2 Situació

A diferència del programa anterior, per tal de calcular de forma eficient la successió de punt es creen unes noves estructures: els diferents valors de les afinitats es guarden en vectors d'una estructura anomenada *Afinity*, determinant la mida del vector segons el nombre d'afinitats de l'IFS corresponent; els punts de l'espai dos-dimensional s'emmagatzemen en una estructura anomenada *Point*; i les caixes o quadrilàters de l'espai que serveixen per envoltar figures utilitzen l'estructura *Box*.

5.3 Detalls d'implementació

El càlcul de la successió de punts és senzill ja que només cal iterar els punts per l'afinitat escollida. Per seleccionar a l'atzar una aplicació de l'IFS, abans de començar es calculen les probabilitats associades a cada aplicació amb l'equació 2.1. No obstant, a diferència del programa anterior en que l'usuari ha d'anar movent-se per l'espai per trobar la figura que s'està generant, en aquest el programa s'encarregarà de centrar-la i mostrar-la de forma adient. Per fer-ho, s'utilitza el càlcul d'una caixa fractal inicial que envolta l'atractor de l'IFS. Per trobar-la, calen unes funcions inicials:

- *Det()*: calcula el determinant de la afinitat de l'IFS que es demana.
- *FixedPoint()*: utilitza la regla de Cramer per calcular el punt fix d'una afinitat donada.
- *MassCenter()*: troba el centre de masses d'un IFS com la mitja ponderada dels punts fixos de les diferents afinitats amb pesos el determinant de cada afinitat.
- *ImagePoint()*: retorna el punt imatge d'un punt donat per una afinitat escollida.
- *InitialFractalBox()*: troba una caixa fractal inicial. Primer de tot, busca el centre de masses. Després, calcula els punts fixos de totes les afinitats i troba totes les imatges d'aquests per les diferents afinitats. Per últim, utilitzant d'origen el centre de masses, genera una caixa fractal que engloba tots els punts trobats.

En suma, en la funció principal del bloc *Compute*, *IFSA_Compute()*, es procedeix al càlcul de la successió de punts i es va actualitzant la caixa fractal quan el nou punt es troba fora de la caixa fractal inicial. Els resultats es guarden en un vector on, cada quatre posicions, es determina el punt trobat (amb la tercera dimensió nul·la) i l'afinitat que s'ha utilitzat per trobar el punt.

Un cop calculada la successió de punts i la caixa fractal, des de la funció *DisplayObjects()* es crida *IFSA_Display()*. Aquesta funció dibuixa la caixa fractal de color gris i la

imatge d'aquesta per cada afinitat segons un color que s'assigna a cada afinitat. També pinta el centre de la caixa fractal, corresponent al centre de la visualització, de color gris; el centre de masses en vermell i el punt fix de cada afinitat amb el color corresponent. Per acabar, utilitza el vector de resultats per dibuixar tots els punts a l'espai, cadascun amb el color de l'afinitat pertinent.

5.4 Utilitats

A més del que s'ha comentat, s'inclouen altres utilitats que permeten extreure més profit del programa. En primer lloc, es pot escollir si veure l'atractor en colors amb les diferents caixes fractals per apreciar l'acció de les diferents afinitats o bé en blanc i negre, ressaltant només l'atractor. En segon lloc, permet modificar amb l'ajuda dels controls els diferents paràmetres de les afinitats de l'IFS. En tercer lloc, es calculen el determinant, els valors propis i els punts fixos de les diferents afinitats i es mostren en pantalla. En quart lloc i per últim, amb el cursor es pot veure quin punt de l'espai s'està observant en cada moment.

5.5 Resultats

Tot plegat, el resultat final és la visualització de l'atractor de l'IFS donat un nombre d'iteracions, de fet amb poques ja es pot començar a observar l'atractor. A més, la modificació dels paràmetres ens deixa variar l'atractor de l'IFS i veure com la dependència contínua dels fractals en els paràmtres es compleix.

Moltes de les figures incloses en la memòria són bons exemples dels resultats que es poden obtenir amb aquest programa. A continuació un exemple en colors permet veure tot el que s'ha explicat:

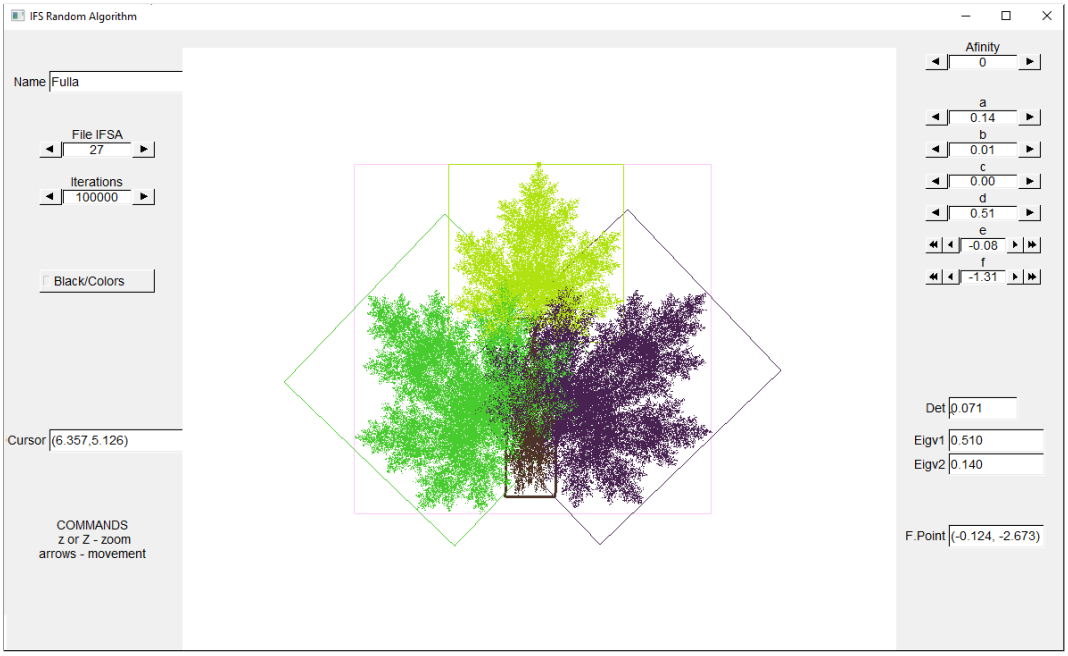

Figura 12: Interfície gràfica del programa *IFS Algoritme Aleatori*.

6 Arbres dos-dimensionals

6.1 Objectiu

L'objectiu d'aquest programa és generar arbres fractals dos-dimensionals a partir de la combinació de l'algoritme determinista i els IFS amb condensació. La base d'aquest serà doncs el programa *IFS Algoritme Determinista* modificat per ser eficient i treballar correctament amb els IFS amb condensació. Recordem que l'única diferència entre aquests IFS i els que hem vist abans és que aquests inclouen una transformació de condensació que envia qualsevol punt de l'espai de fractals a un punt de l'espai de fractals, sempre al mateix. En el cas dels arbres fractals, com s'ha comentat anteriorment, aquest punt correspon al tronc inicial.

6.2 Situació

Seguint amb el funcionament que tenia el programa *IFS Algoritme Determinista*, en aquest cas les figures geomètriques dos-dimensionals corresponen a quadrilàters de paràmetres *Radius, Height* (h) i *Reduction* que donen a cada figura forma de tronc o branca. Les aplicacions de l'IFS en forma de matrius 4x4 són calculades automàticament pel programa. En primer lloc, la transformació de condensació correspon a la matriu identitat permetent així dibuixar el tronc inicial com veurem a continuació. Pel que fa a les altres aplicacions caldrà tantes com marqui *Branches*. Cadascuna ha de correspondre a portar el tronc inicial a la part superior d'aquest, escalar-lo per *Factor* (f) i rotar-lo un cert angle marcat per *Angle* i el nombre de branques. És fàcil veure que aquest procediment correspon a una translació vertical, un escalat i una rotació combinades en l'ordre correcte, obtenint la matriu de transformació:

$$\begin{pmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & h \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} f\cos\theta & -f\sin\theta & 0 & 0 \\ f\sin\theta & f\cos\theta & 0 & h \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

on θ varia per a cada aplicació entre -*Angle* i +*Angle* en radians segons el nombre de branques. En aquest cas, també es calcula la recursivitat fins un cert nivell que determina el paràmetre *Level* per evitar la saturació del programa.

6.3 Detalls d'implementació

Fixem-nos ara com es creen els arbres fractals amb aquestes aplicacions creades. Utilitzarem el mateix procés de generar totes les combinacions possibles fins a nivell *Level* de les afinitats de l'IFS incloent la transformació de condensació representada per la identitat. El resultat seria correcte però cal observar que d'aquesta manera es repeteixen càlculs i el programa no és del tot eficient. Posem per exemple un arbre amb $Branch = 2$, pel que tindrem 3 afinitats (Id, Af.1 i Af.2), i $Level = 2$. En aquest arbre una de les branques que sorgeixen del tronc inicial es generaria dos cops, amb Id·Af.1 i també amb Af.1·Id, representant amb · la combinació d'afinitats. Per evitar això, cal tallar la recursivitat d'alguna manera per no generar repeticions. El que es fa és tallar-la cada cop que s'aplica la identitat, així les combinacions que es generarien amb aquest arbre serien Id, Af.1·Id, Af.2·Id, Af.1·Af.1, Af.1·Af.2, Af.2·Af.1, Af.2·Af.2. No obstant, per evitar modificar la

manera amb què es pinten de diferents colors els diferents nivells amb *LEVEL*, aquelles combinacions que es tallen abans de temps s'omplenen amb identitats fins a acabar el nivell màxim, en aquest cas la primera de l'exemple quedaria Id·Id.

Tot això queda representat en la funció de recursivitat *DisplayRIFS(level,fin)* que on level va disminuint de *Level* fins a 0 i *fin* s'inicia en un valor diferent de 0 i marca quan s'ha utilitzat la identitat i només es poden afegir més identitats fins al nivell màxim. Aprofitarem que la matriu identitat és la afinitat número zero per aprofitar el valor de *fin*. Aquesta recursivitat segueix traient profit de les piles de matrius sense perdre eficiència en el càlcul.

1. Ens preguntem si $level == 0$. Si és així passem al punt 2, sinó al punt 3.
2. Dibuixem el tronc o branca amb les mides adients.
3. Si *fin* no és zero, per a tantes afinitats com tinguem incloent la identitat: copiem la matriu, la multipliquem per la afinitat seleccionada, cridem *DisplayRIFS(level-1,i)* on *i* marca el número d'afinitat i en acabar esborrem la matriu. Si *fin* és zero, només fem aquest procés per la afinitat número zero, corresponent a la identitat.

6.4 Utilitats

Les utilitats implementades són molt semblants a les que ja s'han vist als programes anteriors. Es poden pintar de diferents colors els resultats d'aplicar les diferents aplicacions de l'IFS en el nivell que marca el paràmetre *LEVEL*, també es permet dibuixar les diferents figures que resulten de variar el nivell de recursivitat amb el botó "Levels" i moure-les amb els controls anomenats *horizontal* i *vertical*, així com veure els resultats en blanc i negre amb el botó "Black/Colors".

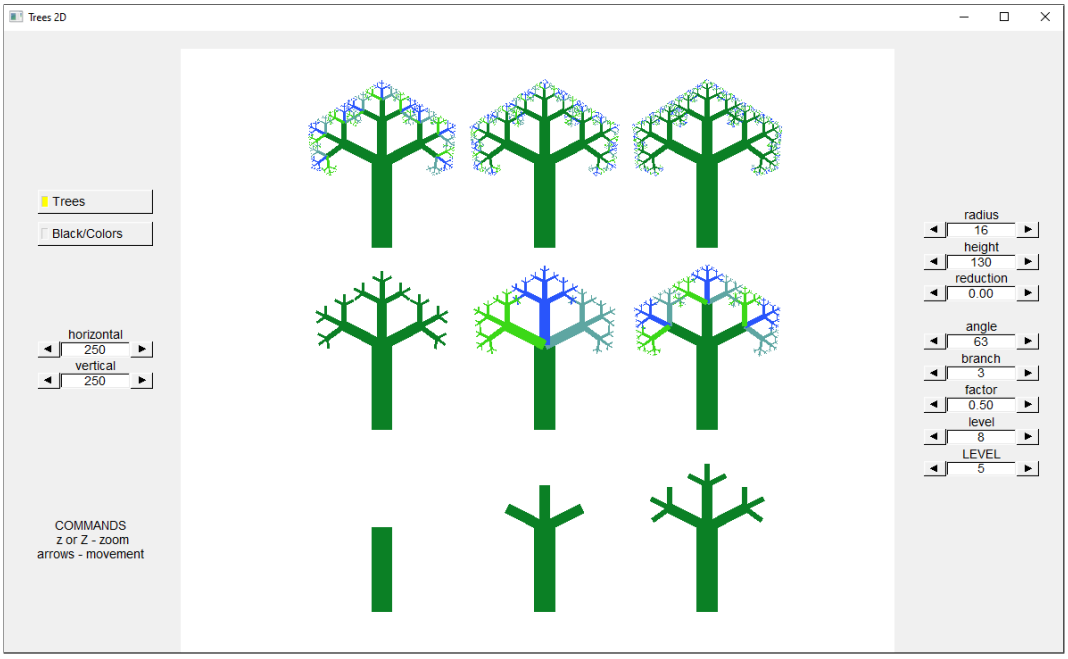

Figura 13: Interfície gràfica del programa *Arbres dos-dimensionals*.

6.5 Resultats

En resum, aquest programa ens permet generar arbres fractals dos-dimensionals d'una forma senzilla i treballar amb els IFS amb condensació. A més, motiva el programa següent en fer el pas de l'espai dos-dimensional a l'espai tres-dimensional per donar un grau més elevat de realisme als objectes finals creats.

Alguns dels exemples que es poden obtenir amb aquest programa són tots els arbres dos-dimensionals vistos al llarg de la memòria així com l'exemple de la Figura 13.

7 Arbres tres-dimensionals

7.1 Objectiu

L'objectiu d'aquest programa és generar arbres fractals en tres dimensions. Per fer-ho, se segueix un mecanisme recursiu amb l'ajuda de les piles de matrius. Es fa tant una generació recursiva d'arbres sense aleatorietat com una generació recursiva d'arbres amb aleatorietat. El tipus de generació que es duu a terme es decideix en la funció *DisplayTree()* contenida en *DisplayObjects()*.

7.2 Generació recursiva d'arbres sense aleatorietat

7.2.1 Situació

En aquest apartat generarem arbres recursius sense aleatorietat. La funció encarregada d'iniciar la recursivitat s'anomena *NormalDisplayBranches()*.

Definim l'arbre a partir d'una estructura geomètrica senzilla que li dóna forma: cons truncats o troncs de con per simular el tronc i les branques, i esferes per simular les fulles i les unions entre branques. Així doncs, queda definit per una sèrie de paràmetres: *Height, Radius, Reduction* defineixen l'altura, el radi inferior i la reducció entre el radi inferior i superior per a cada tronc o branca, *Angle* marca la inclinació en que creixen les noves branques respecte l'anterior, *Factor* determina la relació de mida entre una branca i la següent, *Branch* estableix el nombre de noves branques que es generen cada cop i *Level* marca el nivell de recursivitat màxim.

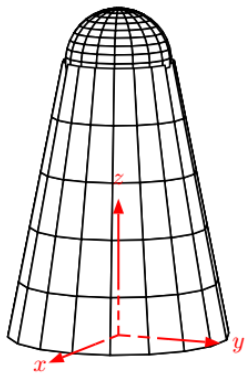

Figura 14: Estructura bàsica de cada tronc o branca.

7.2.2 Detalls d'implementació

Abans de determinar com es crea l'arbre, és necessari crear dues funcions que construeixin les estructures geomètriques bàsiques: *DisplayBranch(Height, Radius, Reduction)* i *DisplayUnionOrLeaf(iter, Radius, Reduction)*. Ambdues treballen de forma similar. Primerament fan una còpia de la matriu de treball perquè els canvis d'aquesta funció no modifiquin la matriu fora de la funció. Llavors es duu a terme l'escalat adient perquè la figura a crear tingui mides unitàries en el nou sistema de coordenades escalat. Seguidament, utilitzant la representació paramètrica de cada figura, es crea el conjunt de vèrtexs

que li donen forma (establint la normal i la coordenada de textura de cadascun). Per últim, s'elimina la matriu i amb aquesta l'escalat que es fa al principi. D'aquesta manera, el resultat final és la figura que volem amb centre la posició inicial i mides les donades. Finalment afegir que el paràmetre *iter* serveix per avisar quan s'està a l'últim nivell i cal crear fulles i no unions entre branques, per la qual cosa cal canviar la textura i crear una esfera una mica més gran.

Les equacions paramètriques permeten també la projecció de la imatge de textura sobre la figura. En el cas tractat de les dues figures unitàries corresponen a les següents:

$$r_{\text{esfera}}(u,v) = (\cos u \cos v, \sin u \cos v, \sin v) \quad \text{on } u \in [0, 2\pi), v \in [-\pi/2, \pi/2)$$
$$r_{\text{con\_truncat}}(u,v) = ((1 - vf)\cos u, (1 - vf)\sin u, v) \quad \text{on } u \in [0, 2\pi), v \in [0, 1)$$

Un cop definides aquestes funcions, definim la funció de recursivitat *BasicDisplay-Branches(iter)* començant per *iter=Level* de la forma següent:

1. Cridem la funció *DisplayBranch(Height, Radius, Reduction).*
2. Ens traslladem a la part superior del tronc fent una translació vertical (respecte a les coordenades del model) de mida *Height.*
3. Cridem la funció *DisplayUnionOrLeaf(iter, Radius, Reduction).*
4. Fem una còpia de la matriu i seguidament un escalat amb el factor d'escala *Factor.*
5. Per a tantes branques com determini *Branch*: copiem la matriu, fem una rotació segons *Angle* i l'angle equiangular determinat pel nombre de branques, cridem *DisplayBranches(iter-1)* si *iter> 0* i en acabar esborrem la matriu d'aquest mateix punt.
6. Esborrem la matriu del punt 4.

Un dels punts més importants és entendre que els paràmetres no varien al llarg de l'arbre, sinó que varia l'escala del sistema de coordenades del model (determinada per la matriu de treball), la qual va patint escalats al punt 4 de cada iteració. A més, el sistema de piles de matrius permet que aquest canvi d'escala es mantingui només en les transformacions que ens interessa.

Com a observació final d'aquest apartat comentar que, per evitar que l'inici de la subbranca sigui més gran que el final de la branca anterior, cal establir sempre la relació entre *Reduction* i *Factor* següent:

$$(1 - Reduction) \cdot Radius \leq Factor \cdot Radius \Rightarrow (1 - Reduction) \leq Factor$$

7.3 Generació recursiva d'arbres amb aleatorietat

7.3.1 Situació

A diferència de l'apartat anterior, en aquest apartat generarem arbres recursius amb aleatorietat introduint variacions aleatòries als diferents paràmetres inicials, que els reanomenarem afegint una "V" (per exemple, $HeightV$). L'única funció que es modifica és la de recursivitat, que ara s'anomena *RandomDisplayBranches().*

Cal fixar-se que, cada cop que es modifica un dels paràmetres inicials amb els controls o bé es varia el punt de vista en la interfície gràfica, l'arbre es recalcula sencer de nou. Aquest fet fa que quan volem introduir variacions aleatòries calgui guardar-les en memòria: en canviar el punt de vista permetrà generar el mateix arbre que teníem i en canviar els paràmetres permetrà recalcular totes les variacions aleatòries per generar-ne un de nou.

7.3.2 Detalls d'implementació

Aquesta reserva de memòria es duu a terme amb l'ajuda de tres vectors: *HeightVvector, FactorVvector* i *RandnumVvector*. Els dos primers vectors utilitzen una distribució uniforme (es podrien utilitzar altres) per generar valors aleatoris decimals entre $(-HeightVariation, HeightVariation)$ i $(0, FactorVariation)$, sent *HeightVariation* i *FactorVariation* dos nous paràmetres en $[0, 1)$ predefinits. El tercer vector genera nombres aleatoris enters qualssevol que serveixen per variar la resta de paràmetres. La mida d'aquests ve donada per la suma de la sèrie geomètrica: $maxLength = 1 + b + b^2 + \cdots + b^l = \sum_{i=0}^{l} b^i = \frac{1-b^{l+1}}{1-b}$ sent $b = Branch$ i $l = Level$, que marca el nombre branques que té l'arbre sense aleatorietat.

Per marcar la posició actual en què es troba la recursivitat dins el vector utilitzem dos índexs *BranchNum* i *subBranchNum*. Es necessiten dos independents per evitar problemes amb el bucle que es duu a terme al punt 5 de la recursió. Com també hi ha aleatorietat afegint branques o nivells es podria donar el cas d'una violació de segment en sortir de l'índex màxim del vector. Per solucionar-ho, es duu a terme el mòdul de *BranchNum* i *subBranchNum* pel valor *maxLength* tornant en aquests casos a l'inici del vector.

Per variar la resta de paràmetres inicials amb els nombres aleatoris enters que conté el tercer vector afegim una sèrie de nous paràmetres: *addBranchProbability, removeBranchProbability, lowBranchProbability* i *addLevelProbability* són el valor màxim d'un dau imaginari que es llença, donant com a resultat el nombre aleatori corresponent mòdul aquest valor màxim, i si aquest és 0 es duu a terme l'acció; *AngleVariation* i *ThetaVariation* marquen la variació de l'angle permesa.

Així doncs, es duen a terme els canvis següents:

- L'altura disminueix o augmenta segons *HeightVariation*.
- El factor augmenta segons *FactorVariation*.
- S'afegeix una branca quan surt un zero segons *addBranchProbability*.
- Es treu una branca quan surt un zero segons *removeBranchProbability*.
- La generació d'una nova branca es duu a terme a l'altura de mig tronc i no a la part superior quan surt un zero segons *lowBranchProbability*.
- S'afegeix un nivell en la branca corresponent quan surt un zero segons *addLevelProbability*.
- L'angle d'inclinació disminueix o augmenta segons *AngleVariation*.
- L'angle equiangular entre branques disminueix o augmenta segons *ThetaVariation*.
- El radi i la reducció no es modifiquen per evitar complicacions amb l'observació final de l'apartat anterior.

Óbviament, aquests són els canvis que s'han implementat per considerar-se els més essencials i assequibles de dur a terme amb el programa fet, però no deixen de ser una mínima part de l'enorme quantitat de variacions que es podrien implementar.

7.4 Utilitats

Aquest programa inclou utilitats diverses que permeten anar modificant constantment els resultats. Inclou un primer botó per escollir quin tipus de generació es duu a terme i un segon per dibuixar o no les fulles en les últimes branques. A més dels controls que modifiquen els paràmetres comentats, també s'inclou un altre que modifica la mida de les fulles. També es mostra en pantalla la longitud i la latitud del punt de vista i es permet moure'l amb els controls. Per últim, s'aprofiten les textures i la il·luminació per donar més realisme a la imatge final, dibuixant un paisatge senzill i permetent canviar les textures per crear fractals naturals diversos.

Figura 15: Interfície gràfica del programa *Arbres tres-dimensionals*.

7.5 Resultats 3D

Vet aquí alguns resultats que es poden obtenir utilitzant el programa, canviant les textures i els paràmetres. És fàcil apreciar la importància de la visualització 3D i com aquesta, juntament amb els resultats referents als fractals coneguts a l'apartat teòric, permeten crear fractals naturals amb un grau elevat de semblança amb la realitat.

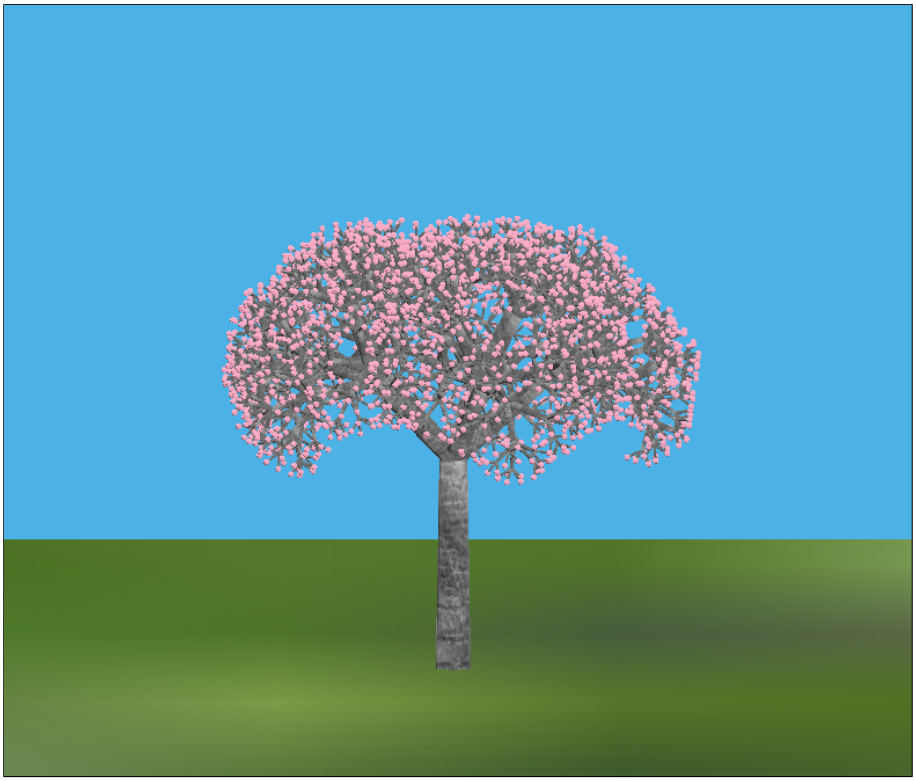

Figura 16: Arbre sense aleatorietat 1 (cirerer).

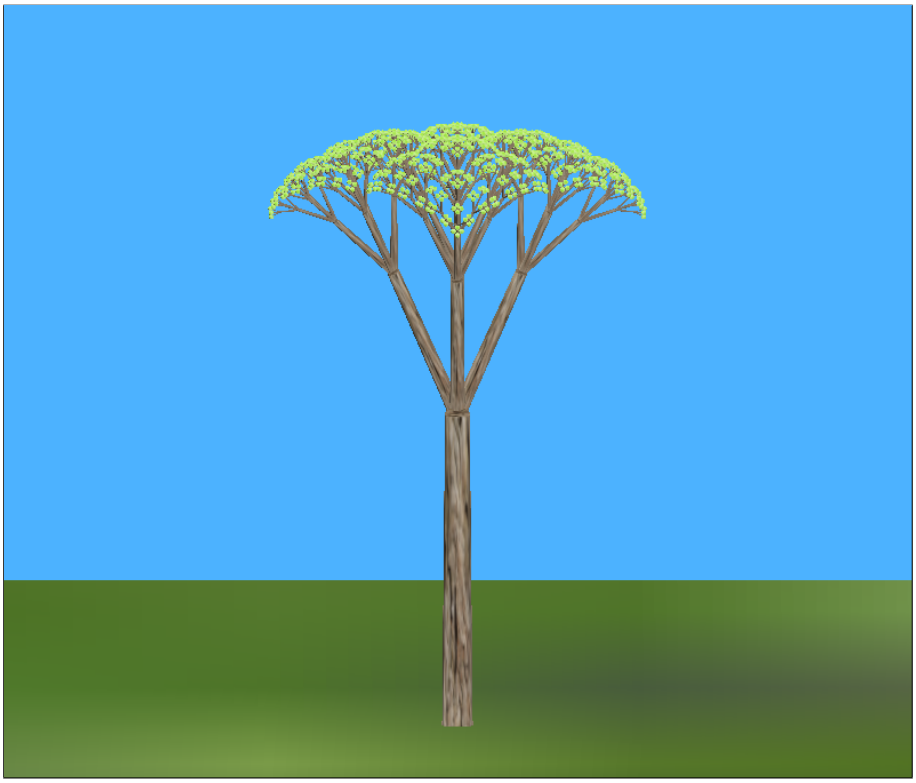

Figura 17: Arbre sense aleatorietat 2.

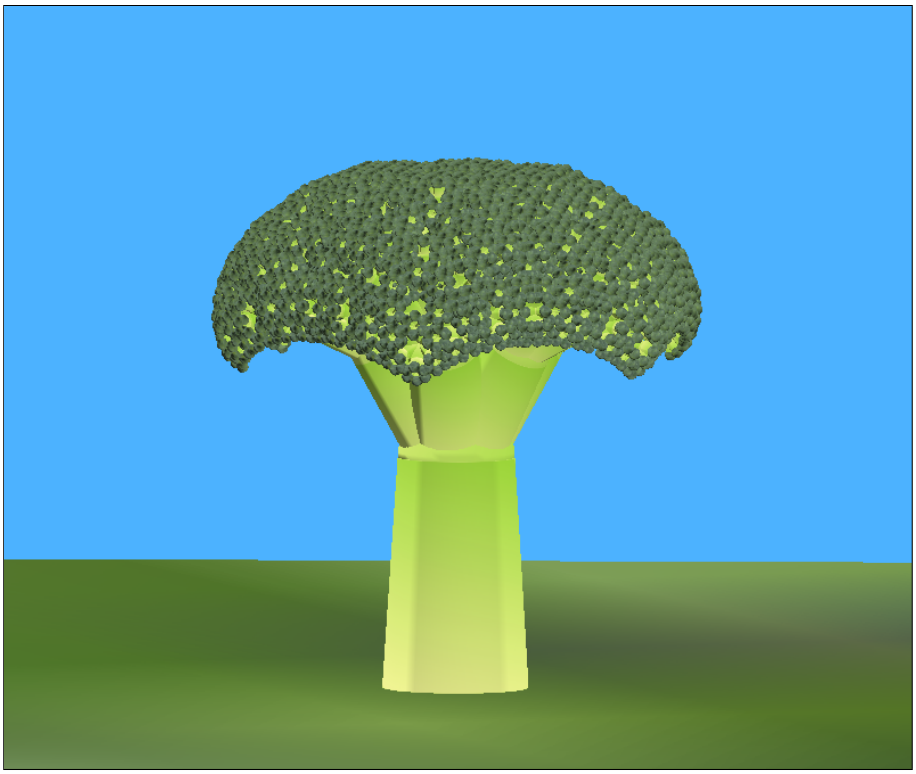

Figura 18: Arbre sense aleatorietat 3 (bròquil).

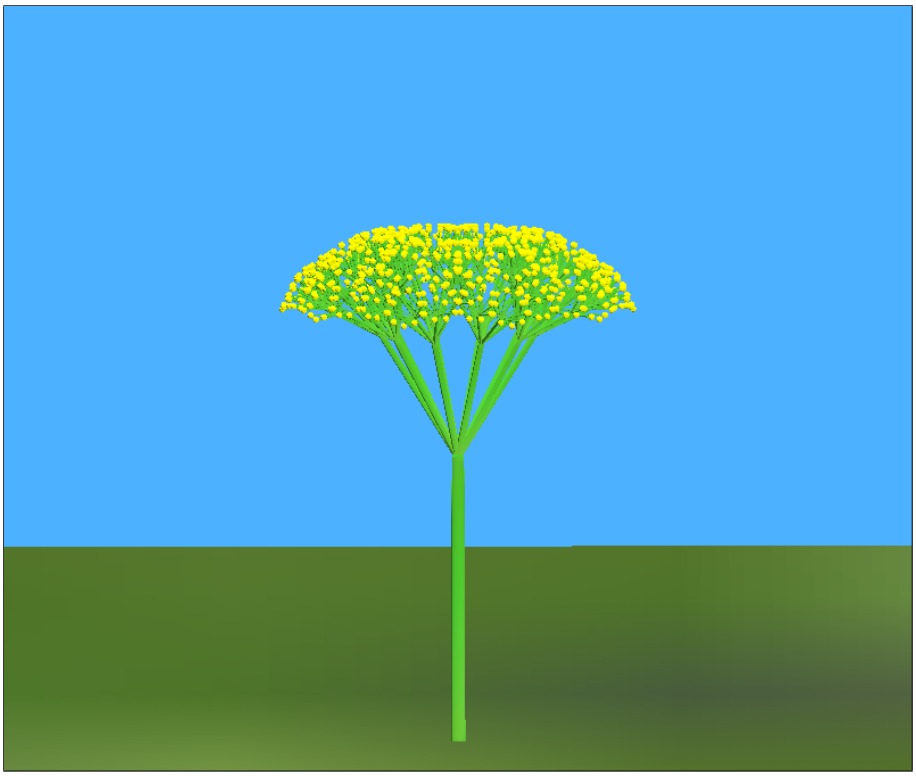

Figura 19: Flor sense aleatorietat 1.

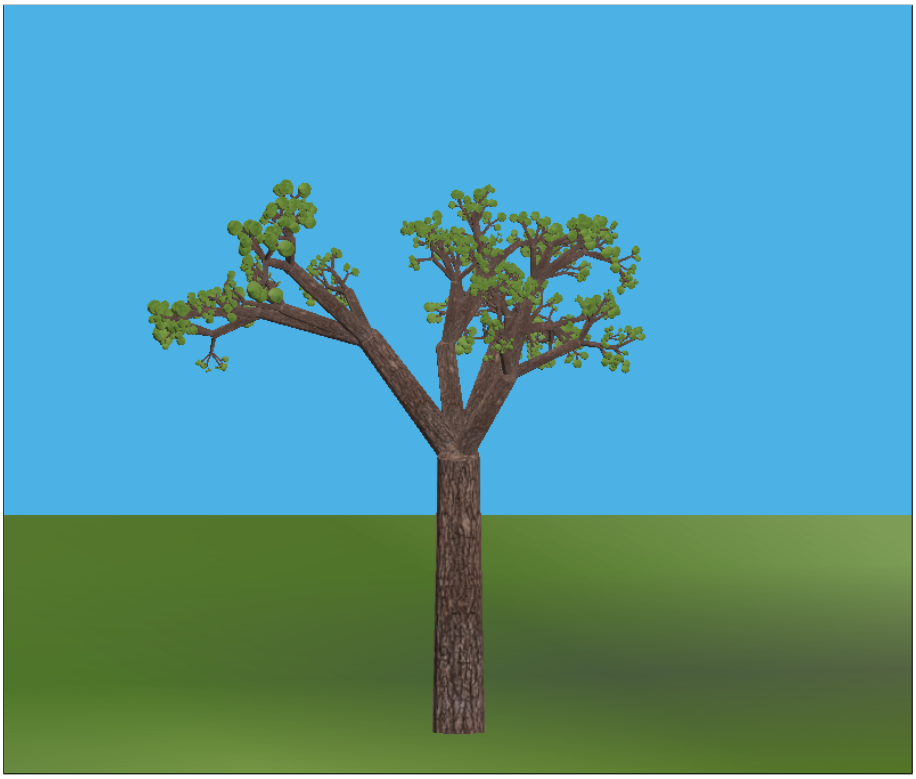

Figura 20: Arbre amb aleatorietat 1.

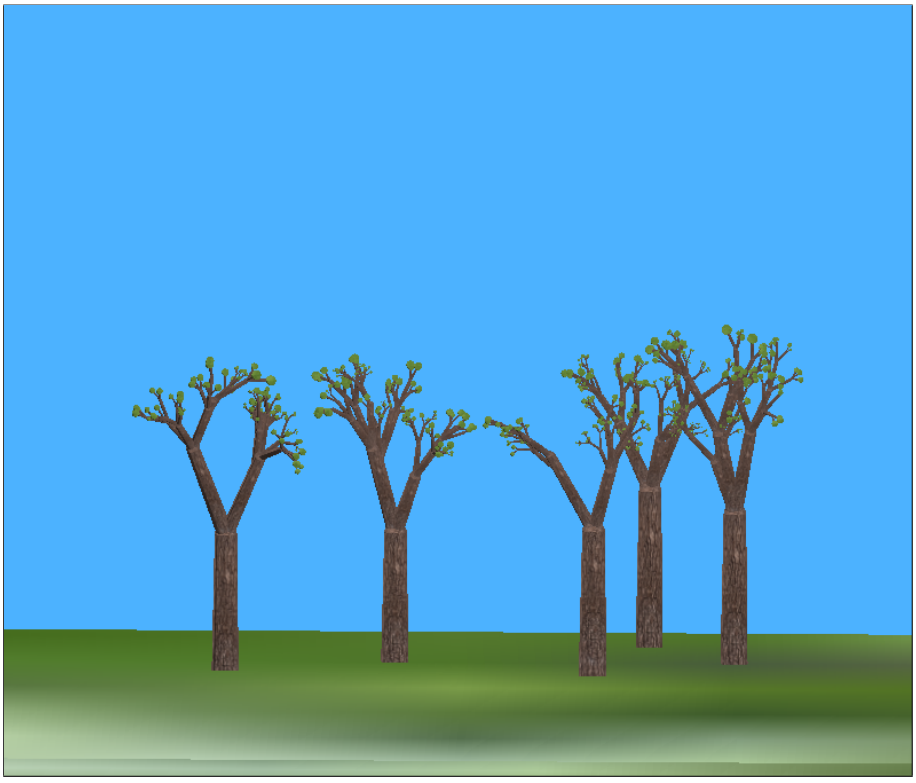

Figura 21: Arbres amb aleatorietat 2.

8 Conclusions

Aquesta memòria constitueix un recull dels principals resultats referents als fractals deterministes i proporciona programes informàtics centrats en la generació d'aquests. Com s'ha pogut observar en la lectura, la part teòrica fa de base de la part pràctica, però alhora els resultats que s'obtenen amb els programes serveixen per anar entenent tots els continguts.

En aquest treball ens hem centrat en aquesta part de la geometria fractal i hem pogut conèixer la seva aplicació. Cal tenir en compte també que hi ha altres tipus de fractals amb altres característiques, com podrien ser els generats per un algoritme d'escapament, com el conegut conjunt de Mandelbrot, o els que es generen amb processos no deterministes, com el moviment brownià. El seu estudi té aplicacions completament diferents.

L'intent de simular la natura de la mà de la geometria fractal ens ha portat a la creació de l'últim programa, centrat en la generació de fractals naturals. Degut a que la naturalesa en sí presenta tants trets i variacions distintes, és complicat simular-les totes. No obstant, el programa permet fer-se una idea de l'estreta relació entre la geometria fractal i la natura. A més, la propera persona lectora pot seguir implementant nous trets per seguir simulant, amb més precisió, arbres de tota mena.

Per l'interès personal en la docència, la motivació per donar a conèixer aquesta geometria no només als estudis superiors i la senzillesa d'aquesta i el seu abast, aquest projecte tindrà sens dubte molt més recorregut donant ús als programes creats.

Els coneixements bàsics que s'han hagut de fer servir, en majoria, han estat treballats en assignatures diverses del grau. Altres s'han aprés durant el desenvolupament del projecte, sobretot amb la utilització de recerca bibliogràfica en format paper per la part teòrica i en format digital per la part pràctica.

Annexes

Conceptes previs

En aquest annex es troben la majora dels conceptes fonamentals que es donen per coneguts en la part teòrica. Està dirigit a l'alumnat que s'inicia per primer cop en una lectura de caire matemàtic. La seva ampliació o demostració es pot trobar en un curs inicial d'Anàlisi Matemàtica.

Definició 8.1. *Un espai $\boldsymbol{X}$ és un conjunt. Els punts de l'espai són els elements del conjunt.*

Definició 8.2. *Un espai mètric $(\boldsymbol{X}, d)$ és un espai $\boldsymbol{X}$ dotat d'una funció $d : \boldsymbol{X} \times \boldsymbol{X} \to \mathbb{R}$, anomenada mètrica, que mesura la distància entre parells de punts x i y de $\boldsymbol{X}$ que compleix:*

(i) $d(x, y) = d(y, x) \quad \forall x, y \in \boldsymbol{X}$

(ii) $0 < d(x, y) < \infty \quad \forall x, y \in \boldsymbol{X}, x \neq y$

(iii) $d(x, x) = 0 \quad \forall x \in \boldsymbol{X}$

(iv) $d(x, y) \leq d(x, z) + d(z, y) \quad \forall x, y, z \in \boldsymbol{X}$

Definició 8.3. *Una funció $f : \mathbf{X}_1 \to \boldsymbol{X}_2$ d'un espai métric $(\boldsymbol{X}_1, d_1)$ en un espai métric $(\boldsymbol{X}_2, d_2)$ és contínua si, per a tot $\epsilon > 0$ i $x \in \boldsymbol{X}_1$, existeix $\delta > 0$ tal que*

$$d_1(x, y) < \delta \Rightarrow d_2(f(x), f(y)) < \epsilon$$

Definició 8.4. *Una successió $\{x_n\}_{n=1}^{\infty}$ de punts en un espai mètric $(\boldsymbol{X}, d)$ s'anomena successió de Cauchy si, per a qualsevol nombre $\epsilon > 0$, existeix un enter $N > 0$ tal que*

$$d(x_n, x_m) < \epsilon \quad \textit{per a tot } n, m > N$$

Definició 8.5. *Una successió $\{x_n\}_{n=1}^{\infty}$ de punts en un espai mètric $(\boldsymbol{X}, d)$ es diu que convergeix a un punt $x \in \boldsymbol{X}$ si, per a qualsevol nombre $\epsilon > 0$, existeix un enter $N > 0$ tal que*

$$d(x_n, x) < \epsilon \quad \textit{per a tot } n > N$$

En aquest cas el punt $x \in \boldsymbol{X}$ al qual convergeix la successió s'anomena límit de la successió i escrivim

$$x = \lim_{n \to \infty} x_n$$

Teorema 8.6. *Siguin $(\boldsymbol{X}, d)$ un espai mètric, $\{x_n\}$ una successió de Cauchy convergent a $x \in \boldsymbol{X}$ (o equivalentment sigui $\{x_n\}$ una successió i x un punt tal que $\lim_{n \to \infty} d(x, x_n) = 0$) i $f : \boldsymbol{X} \to \boldsymbol{X}$ una funció contínua. Aleshores*

$$\lim_{n \to \infty} f(x_n) = f(x)$$

Teorema 8.7. *Si una successió de punts $\{x_n\}_{n=1}^{\infty}$ en un espai mètric $(\boldsymbol{X}, d)$ convergeix a un punt $x \in \boldsymbol{X}$, aleshores $\{x_n\}_{n=1}^{\infty}$ és una successió de Cauchy.*

Definició 8.8. *Un espai mètric $(\boldsymbol{X}, d)$ és complet si tota successió de Cauchy $\{x_n\}_{n=1}^{\infty}$ en $\boldsymbol{X}$ té límit $x \in \boldsymbol{X}$.*

Definició 8.9. *Sigui $S \subset \boldsymbol{X}$ un subconjunt d'un espai mètric $(\boldsymbol{X}, d)$. Un punt $x \in \boldsymbol{X}$ s'anomena punt límit de S si existeix una successió $\{x_n\}_{n=1}^{\infty}$ de punt $x_n \in S \setminus \{x\}$ tal que $\lim_{n\to\infty} x_n = x$.*

Definició 8.10. *Sigui $S \subset \boldsymbol{X}$ un subconjunt d'un espai mètric $(\boldsymbol{X}, d)$. La clausura de S, denotada per $\bar{S}$, es defineix com $\bar{S} = S \cup \{\text{punts límit de } S\}$. S és tancat si conté tots els seus punts límit, és a dir, $S = \bar{S}$.*

Proposició 8.11. *Sigui S un subconjunt d'un espai mètric complet $(\boldsymbol{X}, d)$. Aleshores (S, d) és un espai mètric. A més, (S, d) és complet si, i només si, S és tancat en $\boldsymbol{X}$.*

Definició 8.12. *Sigui $S \subset \boldsymbol{X}$ un subconjunt d'un espai mètric $(\boldsymbol{X}, d)$. S és compacte si tota successió infinita $\{x_n\}_{n=1}^{\infty}$ en S conté una subsuccessió que té límit en S.*

Definició 8.13. *Sigui $S \subset \boldsymbol{X}$ un subconjunt d'un espai mètric $(\boldsymbol{X}, d)$. S és fitat si existeix un punt $a \in \boldsymbol{X}$ i un nombre $R > 0$ tal que*

$$d(a, x) < R \quad \forall x \in S$$

Definició 8.14. *Sigui $S \subset \boldsymbol{X}$ un subconjunt d'un espai mètric $(\boldsymbol{X}, d)$. S és totalment fitat si, per a cada $\epsilon > 0$, existeix un conjunt finit de punts $\{y_1, y_2, \ldots, y_n\} \subset S$ tal que quan $x \in S$, $d(x, y_i) < \epsilon$ per a algun $y_i \in \{y_1, y_2, \ldots, y_n\}$. Tal conjunt de punts $\{y_1, y_2, \ldots, y_n\}$ s'anomena ϵ-xarxa.*

Teorema 8.15. *Sigui $S \subset \boldsymbol{X}$ un espai mètric complet. Sigui $S \subset \boldsymbol{X}$. Aleshores S és compacte si, i només si, S és tancat i totalment fitat.*

Definició 8.16. *Sigui S un conjunt de nombres reals. Denotem per*

$$\inf S = \max\{x \in \mathbb{R} : x \leq s \text{ per a tot } s \in S\}$$

$$\sup S = \min\{x \in \mathbb{R} : x \geq s \text{ per a tot } s \in S\}$$

l'ínfim i el suprem d'un conjunt S de nombres reals. Per la naturalesa dels nombres reals, aquests sempre existeixen.

Il·luminació i textures

La il·luminació és un punt molt important per donar realisme a les visualitzacions 3D, en concret al programa d'arbres tres-dimensionals. *OpenGL* calcula la il·luminació en cada vèrtex del model i obté una terna RGB que s'utilitza per donar color als polígons que forma. El valor obtingut a cada vèrtex depèn de les característiques de les fonts de llum definides així com les característiques del material assignat al polígon que es dibuixa. Així doncs, alguns dels factors que intervenen en el càlcul són:

- **Habilitar el mode d'il·lumincació i crear les fonts de llum**: mitjançant *glEnable(GL_LIGHTING)* per activar el mode i *glEnable(GL_LIGHTi)* que permet posar fins a vuit fonts de llum.
- **Determinar el tipus de llum**: direccional, on la llum ve d'una direcció i tots els rajos de llum són paral·lels, posicional, on la llum està posicionada en un punt concret de l'espai i il·lumina en totes les direccions, o focal, on la llum emesa només il·lumina en una direcció en forma de con.
- **Modificar les components de la llum**: es caracteritza la llum amb diferents components (ambient, difusa, especular o brillantor) que li donen diferents característiques i es modifiquen mitjançant la funció *glLightfv()*.
- **Activar la llum ambient**: aquella que no prové de cap font de llum en particular, sinó que és inherent a l'escena. S'utilitza la funció *glLightModelfv()*.
- **Posicionar l'observador/a**: per defecte, i en el nostre cas, se situa l'observador/a a l'infinit, pel que la direcció entre els vèrtexs a il·luminar i l'observador/a és la mateixa per a tots els objectes de l'escena.
- **Establir les propietats dels materials**: la forma amb la qual la llum incideix sobre les superfícies dels objectes depèn de les propietats del material dels mateixos. Aquestes es defineixen mitjançant la funció *glMaterialfv()* que permet determinar tant la cara de l'objecte per la qual s'aplica el material com les propietats del material, semblants a les de la llum: ambient, difusa, especular i brillantor, principalment.
- **Calcular les normals als vèrtexs**: les superfícies s'il·luminen de forma diferent depenent de l'angle d'incidència de la llum. Com més perpendicular sigui la llum a la superfície més s'il·luminarà el vèrtex. Aquest angle d'incidència es calcula mitjançant la normal.
- **Determinar el mode de renderitzat**: un cop fets tots els càlculs per a cada vèrtex, es pot decidir si omplenar cada polígon escollint només el resultat d'un dels vèrtexs o bé fent una combinació interpolant els resultats dels vèrtexs que el formen. S'utilitza la funció *glShadeModel()*.

També les textures permeten donar un gran realisme als nostres objectes. Treballarem en el nostre cas amb textures d'imatge, que emmagatzemen els valors en una imatge 2D. És necessari indicar com volem aplicar aquesta textura 2D a la geometria del model 3D, és a dir, cal especificar la projecció que indiqui com es recobrirà l'objecte amb el mapa de textura. Algunes de les funcions més interessant al respecte són: *glBindTexture()* i *glGenTexture()* que permeten treballar amb diferents textures o *glTexCoord2()* que permet establir les coordenades de textura amb les que treballar.

Referències

[1] Barnsley, Michael: *Fractals Everywhere*, Academic Press INC., London, 1988.

[2] Mandelbrot, Benoît: *La geometria fractal de la naturaleza*, Traducció: Josep Llosa, Tusquets Editores S.A., Barcelona, 1997.

[3] González Morcillo, Carlos; A. Albusac Jiménez, Javier; Mora Castro, César; Fernández Durán, Sergio: *Desarrollo de Videojuegos: Programación Gràfica* [llibre en línia], LibroVirtual.org. Disponible en
`http://www.cedv.es/descargas/M2_ProgramacionGrafica_2Ed.pdf`

[4] Espigulé Pons, Bernat: *Fractal Trees* [pàgina web] IWR Heidelberg University, 2012 [consulta: 12 de març de 2020]. Disponible en
`http://pille.iwr.uni-heidelberg.de/ fractaltree01/`

[5] Ho Ahn, Song: *Songho.ca: OpenGL* [pàgina web] edició pròpia, 2005 [consulta: 19 d'abril de 2020]. Disponible en
`http://www.songho.ca/opengl/index.html`

[6] Toal, Ray: *Iterated Function Systems* [pàgina web] Loyola Maryount University, 2008. [consulta: 19 de maig de 2020]. Disponible en
`https://cs.lmu.edu/ ray/notes/ifs/`