Facultatea de
Matematică și Informatică
Universitatea din București

Programarea Aplicațiilor de Simulare

Generarea Procedurală de Conținut

Curs 5

Pătrânjel David-George

Agenda Cursului

01

Generarea Procedurală

02

PCG în Jocuri Video

03

Gramatici

04

Hărți de înălțime

01

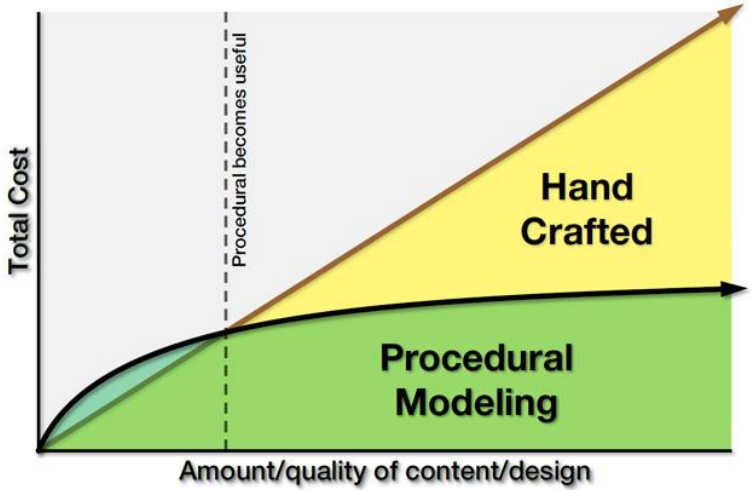
Generarea Procedurală

Generarea Procedurală

- en. Procedural (Content) Generation (PCG)
- **Crearea de conținut într-un mod algoritmic.**
- Combină puterea computațională cu aportul limitat al utilizatorului.
- La ce ne referim prin „conținut”?
  - Hărți și Terenuri
  - Nivele, Misiuni și Dialoguri
  - Texturi și Muzică
  - Obiecte, Arme

De ce folosim PCG?

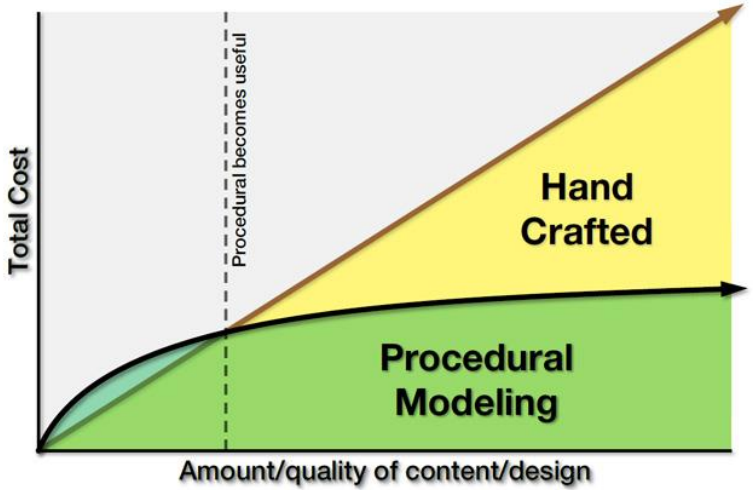
1. Reducerea costurilor
   - PCG poate fi mai avantajoasă decât crearea manuală de conținut.
2. Diversitatea conținutului
   - Generarea de conținut unic pentru
3. Compresia datelor
   - Prea mult conținut pentru a fi stocat.



https://www.e-education.psu.edu/geogvr/node/534

De ce folosim PCG?

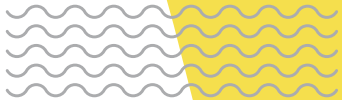
1. **Reducerea costurilor**
2. **Diversitatea conținutului**
3. **Compresia datelor**
4. Dorința de a rejuca jocurile
5. Eficiența de timp
6. Modularitate
7. Unicitate
8. Hărți infinite (Scalabilitate)
9. Aplicarea și respectarea regulilor
10. Artă și creativitate



https://www.e-education.psu.edu/geogvr/node/534

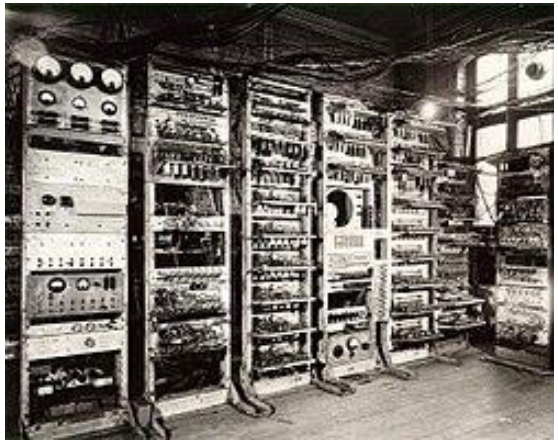
02

PCG în Jocuri Video



Istoria PCG

- 1952 - Christopher Strachey a **creat algoritmul de generare de scrisori de dragoste.**
- Funcționa pe Manchester Mark 1.



1. Print two words taken from a list of salutations
2. Do the following 5 times:
    1. Choose one of two sentence structures depending on a random value Rand
    2. Fill the sentence structure from lists of adjectives, adverbs, substantives, and verbs.
3. Print the letter's closing[9]

Darling Sweetheart,

You are my avid fellow feeling. My affection curiously clings to your passionate wish. My liking yearns for your heart. You are my wistful sympathy: my tender liking.

Yours beautifully

M. U. C.

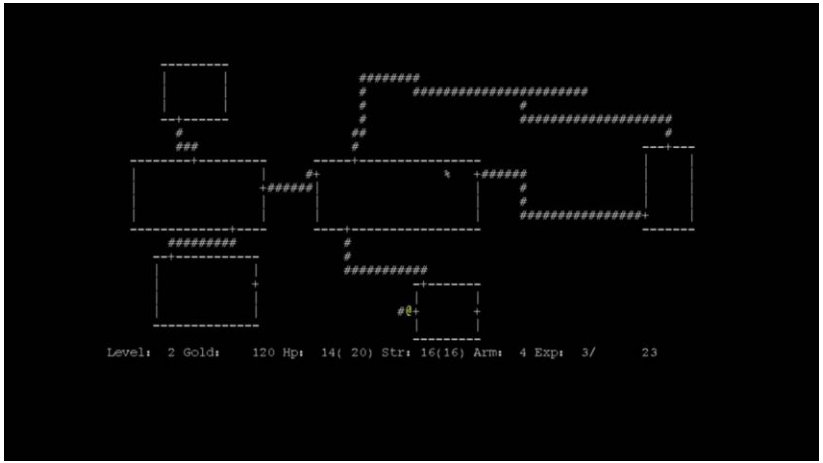
Strachey love letter algorithm

Istoria PCG

- Motivația principală: Compresia datelor.
- Diversitatea jocului provenea din generarea datelor la runtime.

Elite (1984)

8 galaxy cu câte 256 planete



Rogue (1980)

Se generau temințele

Istoria PCG

- În anii 90 au apărut jocurile bazate pe **strategia 4x**.
- **“Explore, Expand, Exploit, and Exterminate"**
- Era nevoie de PCG pentru a genera hărți, asigurând atracția jucătorilor asupra jocurilor.

Civilization 1 (1991)

Istoria PCG

- În cazul jocurilor tip Rogue, se utiliza PCG pentru a genera temnițe și obiecte în mod aleator.
- Ulterior se utiliza PCG în jocuri RPG sau în jocuri desfășurare în spațiul intergalactic.

Diablo (1995)

Spore (2008)

- **Utilizează PCG în multiple arii.**
- Se generează:
  - Caracterele personajelor
  - Terenul și Biomes
  - Flora și Fauna
  - Glaxii
  - Oraște

Spore (2008)

https://www.youtube.com/watch?v=zi2GvqboQfY

Minecraft (2011)

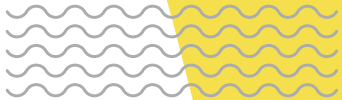
- Dezvoltat de Mojang.
- Utilizează mai multe hărți de zgomot pentru generarea de biomes, în funcție de vreme și altitudine.
- Se generează:
  - Harta infinită din voxeli
  - Păduri
  - Biomes

No Man’s Sky (2016)

- Dezvoltat de Hello Games.
- Sloganul “Every atom procedural”.
- Se generează:
  - Flora și fauna
  - Ecosistem
  - Peisaje
  - Nave spațiale și structure extraterestre
  - **Peste 18 quintilioane de planete unice**

03

Gramatici



Generarea Artei

- Ce este arta?

- Dex: Activitate a omului care are drept scop producerea unor valori estetice și care folosește mijloace de exprimare cu caracter specific.

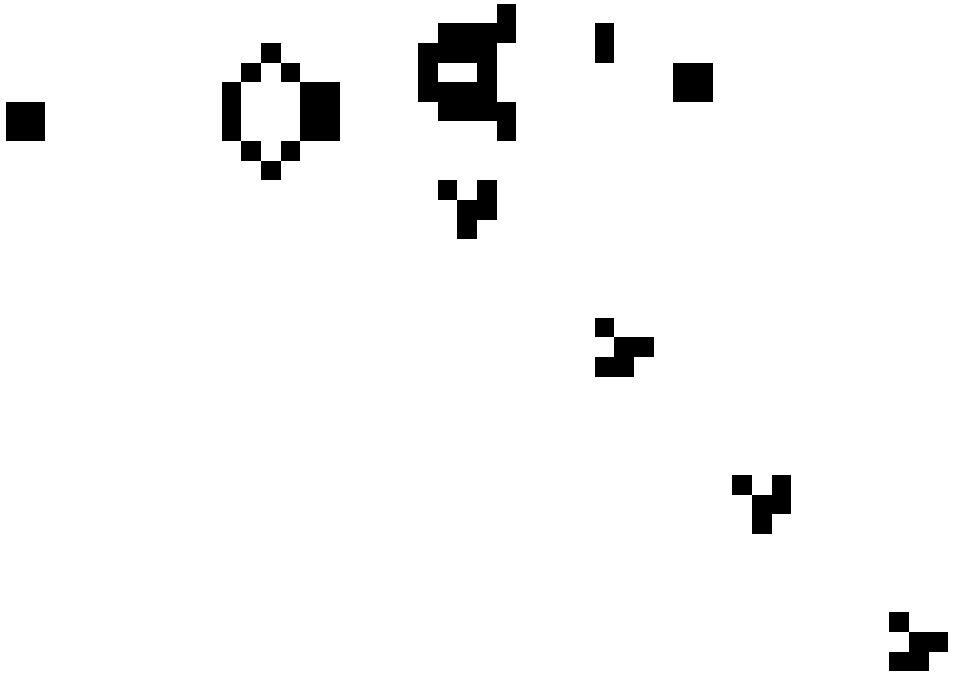
- Putem genera artă prin utilizarea PCG de teren, hărți, caractere și modele, grafici ș.a.m.d.

- Deși intervenția umană este indirectă, aceasta există!

https://www.moma.org/artists/4675-jackson-pollock

Automate Celulare

- Introduse de matematicienii John von Neumann and Stanislaw Ulam (1940).
- Model matematic de reprezentare a sistemelor biologice și a celor de auto-reproducerea biologică.
- Formă de **GRILĂ** de dimensiuni finite.
- Celulele se numesc și **ATOMI** și pot avea diferite forme.
- Atomii se pot regăsii într-o anumită **STARE**, de obicei ON/OFF.
- Atomii învecinați își pot afecta reciproc starea pe baza unor **REGULI**.
- Atomii își schimbă starea de la o generație la altă.
- Putem avea și automate celulare de o dimensiune.
- Joc cu 0 jucători: *Conway's Game Of Life*

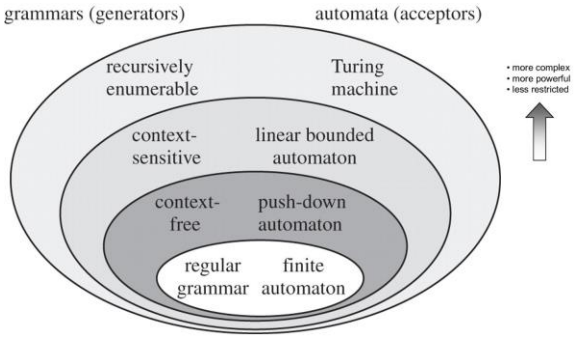


Conway's Game of Life

https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life#/media/File:Gospers_glider_gun.gif

Gramatici

- **Limbajul:** descris prin simboluri și sintaxă – reguli prin care se pot combina simbolurile.
- **Gramatică**: Modalitatea de obținere de șiruri într-un limbaj prin reguli de producție pornind de la un simbol de start.
- **G = (V, T, R, S)**
  - Variabile (A, B, C)
  - Terminale (a, b, c)
  - Reguli (A -> AB)
  - Simbol de start (S)



https://devopedia.org/chomsky-hierarchy

Ierarhia Chomsky

Tip 0: Gramatici nerestricționate

Tip 1: Gramatici dependente de context

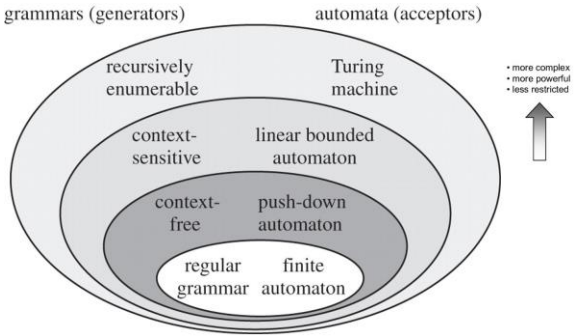
- Reguli αAβ -> αγβ
- Restricție: |αAβ| <= |αγβ|

Tip 2: Gramatici independente de context

- Reguli A -> γ (din variabilă în șir arbitrar)

Tip 3: Gramatici regulate

- Generează limbaje regulate (conversie la DFA)
- Reguli A -> a, A->ε, A -> aB (la dreapta) SAU
- Reguli A -> a, A->ε, A-> Ba (la stânga)



https://devopedia.org/chomsky-hierarchy

$$\boldsymbol{S} \underset{1}{\Rightarrow} \boldsymbol{a}B\boldsymbol{S}\boldsymbol{c} \underset{1}{\Rightarrow} aB\boldsymbol{a}B\boldsymbol{S}cc$$
$$\underset{2}{\Rightarrow} aBaB\boldsymbol{abc}cc$$
$$\underset{3}{\Rightarrow} a\boldsymbol{a}\boldsymbol{B}Babccc \underset{3}{\Rightarrow} aaB\boldsymbol{a}\boldsymbol{B}bccc \underset{3}{\Rightarrow} aa\boldsymbol{a}\boldsymbol{B}Bbccc$$
$$\underset{4}{\Rightarrow} aaaB\boldsymbol{bb}ccc \underset{4}{\Rightarrow} aaa\boldsymbol{bb}bccc$$

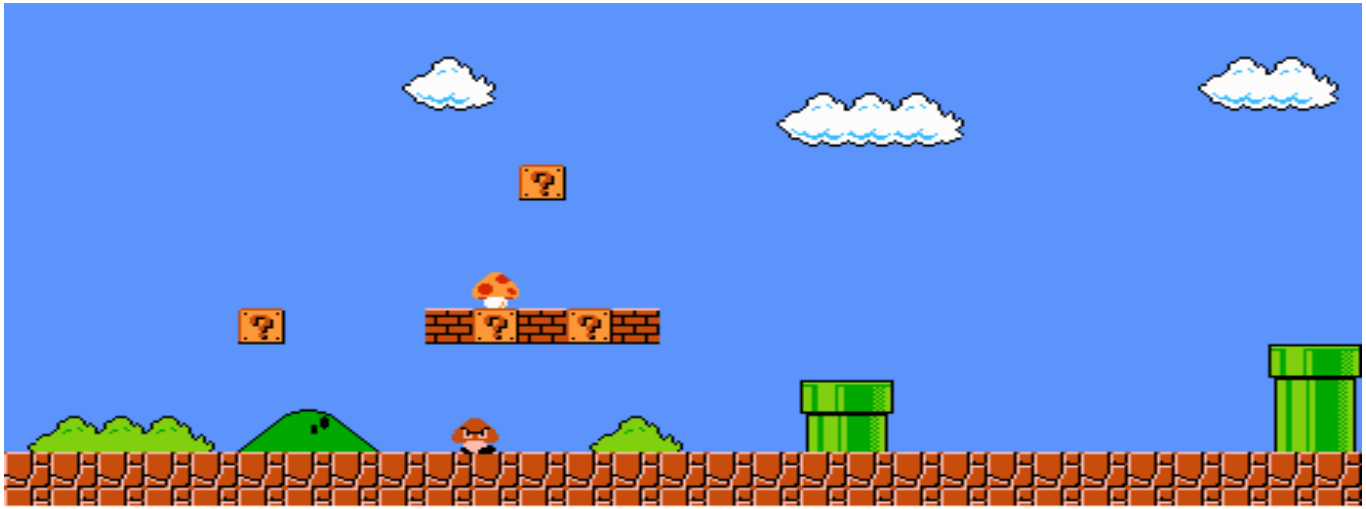
1. $S \to aBSc$
2. $S \to abc$
3. $Ba \to aB$
4. $Bb \to bb$

Exemplu de Gramatică

https://en.wikipedia.org/wiki/Formal_grammar

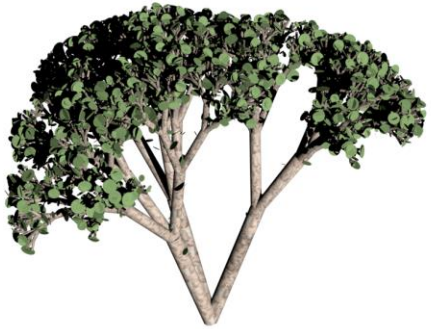
Generarea jocurilor prin gramatici

- Putem reprezenta elementele dintr-un joc prin gramatici.
- Putem interpreta soluția unui nivel ca fiind un cuvânt, având *reguli de joc* și o poziție de start.



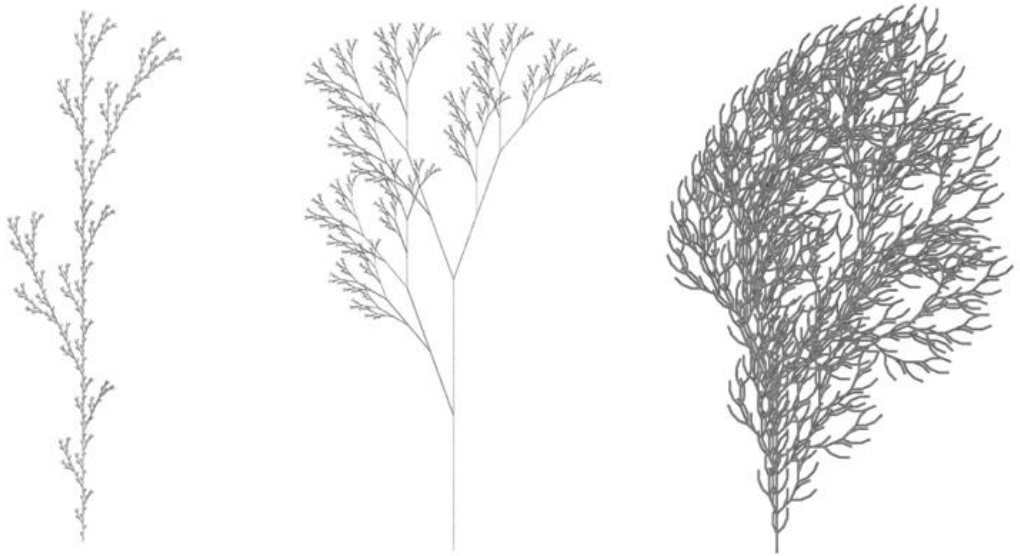
Sisteme Lindenmayer

- Introduse de biologul Aristid Lindenmayer (1968).
- Model matematic de modelare a procesului de creștere a plantelor.
- Utile în PCG în cazul fractalilor și plantelor.
- **Componentele Sistemelor-L:**
  - **Alfabet**: Colecție de simboluri care reprezintă diverse componente ale unei plante.
  - **Axiomă**: Șirul inițial de simboluri.
  - **Reguli de producție**: Colecție de reguli care definesc substituția sau extinderea șirului de simboluri.



Tree with improved realism.

https://allenpike.com/modeling-plants-with-l-systems/



Some 2D plant-like structures from bracketed L-Systems.

Plante Generate cu Sisteme-L (2D)

https://allenpike.com/modeling-plants-with-l-systems/

a
n=5, δ=25.7°
F
F → F[+F]F[-F]F

b
n=5, δ=20°
F
F → F[+F]F[-F][F]

c
n=4, δ=22.5°
F
F → FF-[-F+F+F]+
[+F-F-F]

d
n=7, δ=20°
X
X → F[+X]F[-X]+X
F → FF

e
n=7, δ=25.7°
X
X → F[+X][-X]FX
F → FF

f
n=5, δ=22.5°
X
X → F-[[X]+X]+F[+FX]-X
F → FF

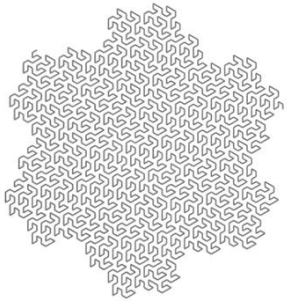
Reguli de Generare

https://paulbourke.net/fractals/lsys/

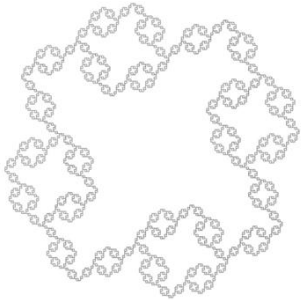
Some determinstic 3D branching plants.

Plante Generate cu Sisteme-L (3D)

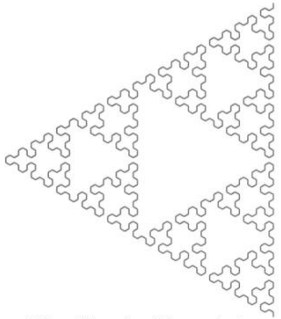
https://allenpike.com/modeling-plants-with-l-systems/



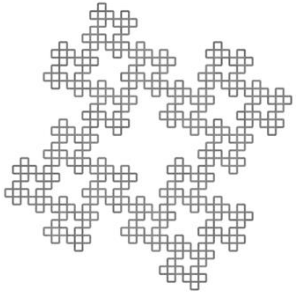
Hexagonal Gopser curve



A Koch curve



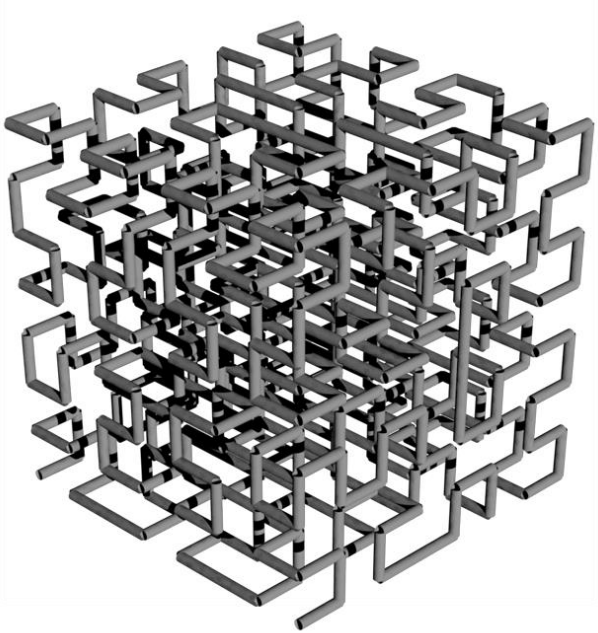
The Sierpinski gasket



A second Koch curve

Fractali folosind Sisteme-L (2D)

https://allenpike.com/modeling-plants-with-l-systems/



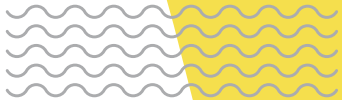
3D Hilbert curve.

Fractali folosind Sisteme-L (3D)

https://allenpike.com/modeling-plants-with-l-systems/

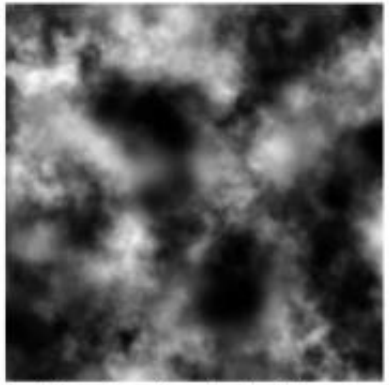
04

Hărți de înălțime

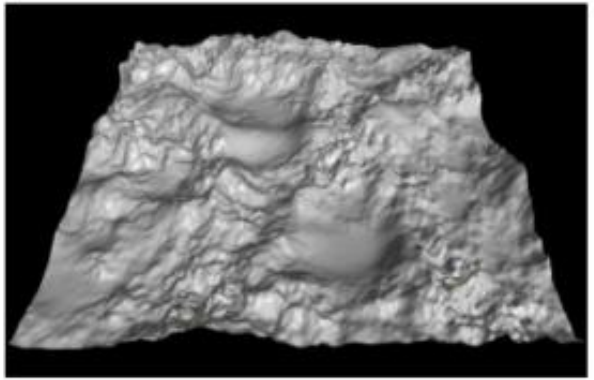


Hărți de înălțime

- Folosind o hartă de înălțime (en. heightmap) putem asocia valorii fiecărui pixel dintr-o imagine o valoare a înălțimii terenului.
- Putem genera un relief diversificat, similar cu cel din lumea reală.
- Nu putem genera peșteri sau prăpastii.

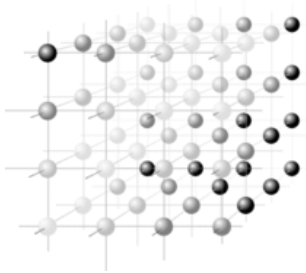


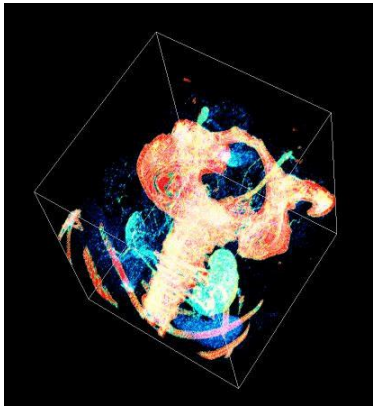
height visualized as brightness

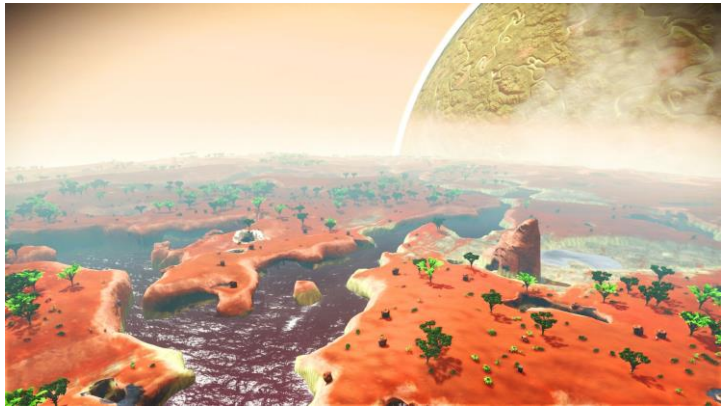


Voxeli

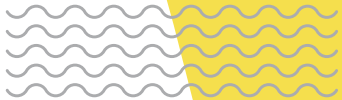
- Voxels = Volume elements
- Putem reprezenta orice forme 3D.
- Necesită un volum mare de date și algoritmi complecși de vizualizare.





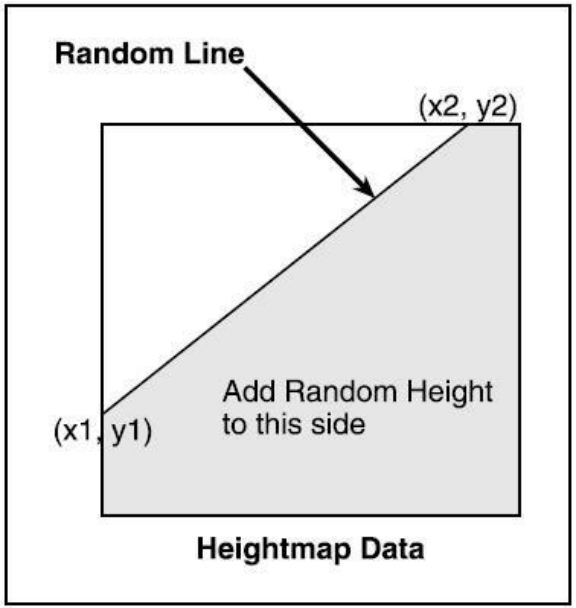


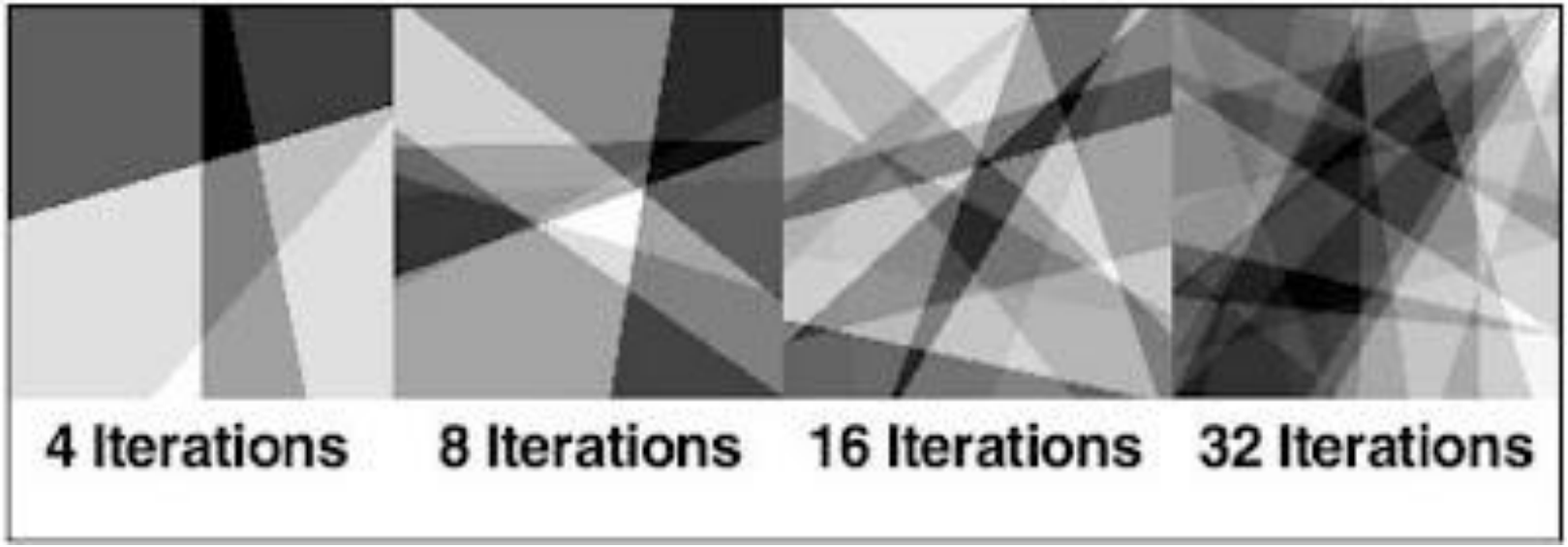
Cum folosim PCG pentru hărți de înălțime?



Algoritmul Fault Formation (FF)

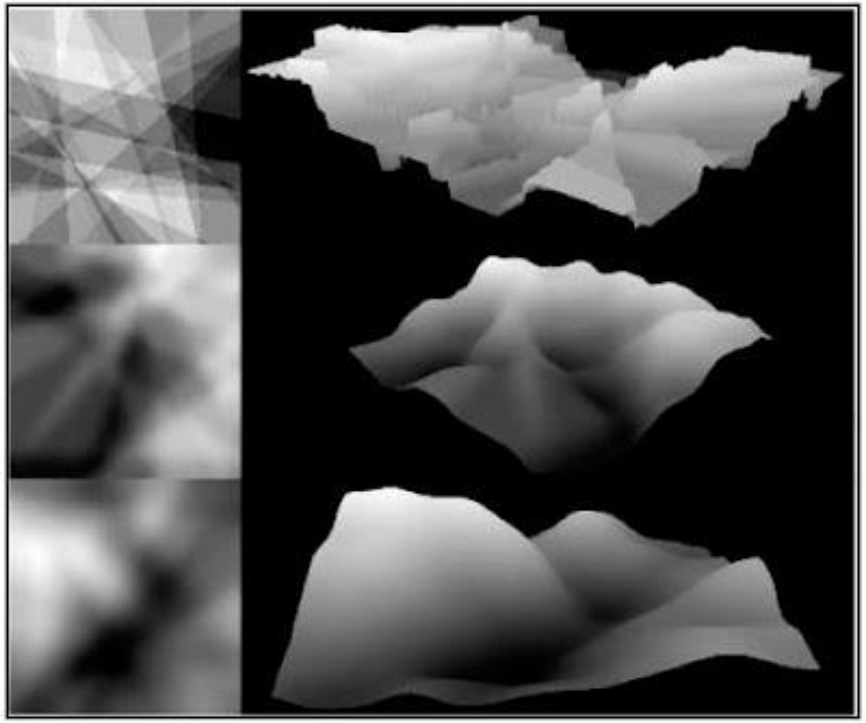
- Folosim o grilă cu înălțimea inițială 0 în fiecare celulă.
- Trasăm o linie aleană în mod aleator.
- Pixelii aflați de o parte (i.e. dreapta) a liniei trasate se vor înălța cu o valoare aleasă.
- Se repeat algoritmul iar valoarea de înălțare a terenului se micșorează la fiecare pas.





Alforitmul de Fault Formation

https://jordanliuhao.blogspot.com/2007/09/fractal-terrain-generation-fault.html

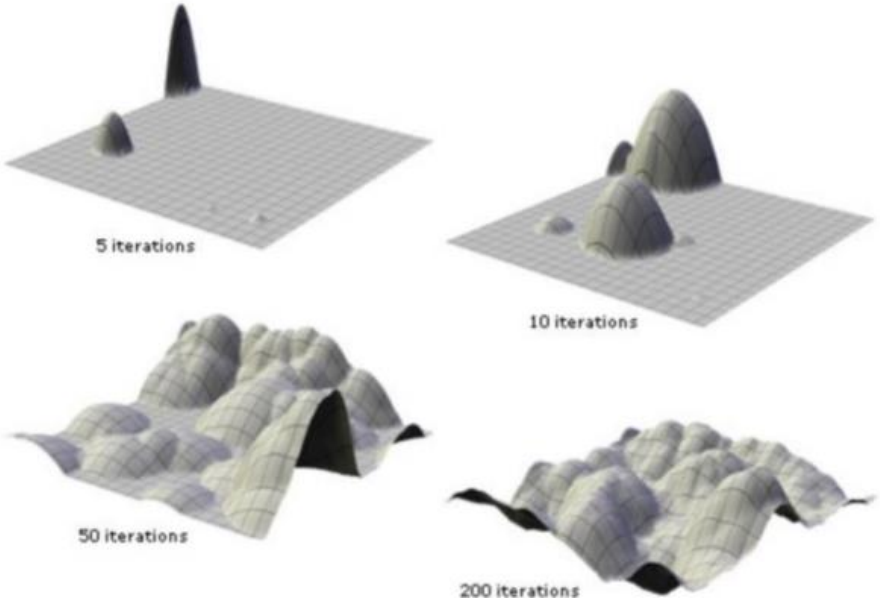


Teren folosind FF și filtru de eroziune

https://jordanliuhao.blogspot.com/2007/09/fractal-terrain-generation-fault.html

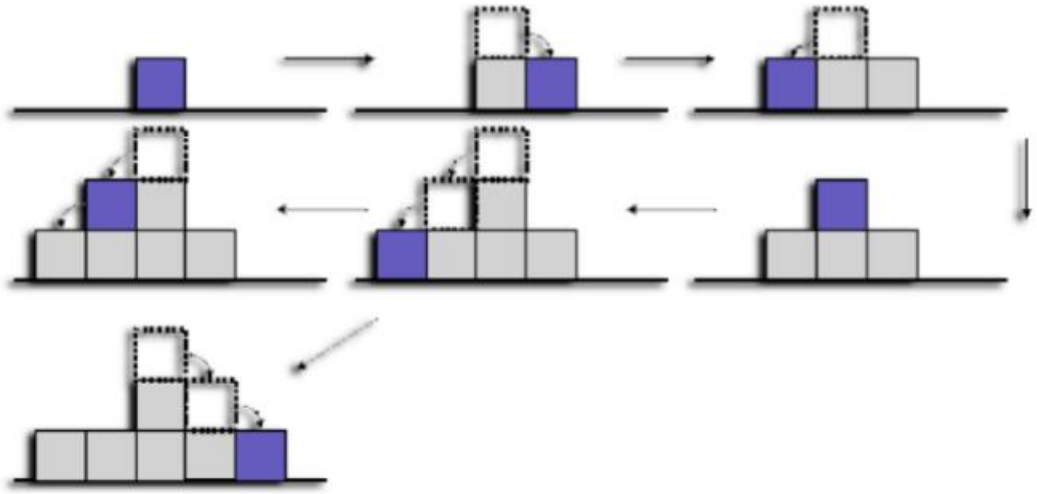
Adăugarea de dealuri

- Folosim o grilă cu înălțimea inițială 0 în fiecare celulă.
- Se adaugă la fiecare pas un deal (parabolă) de înălțime aleatoare.



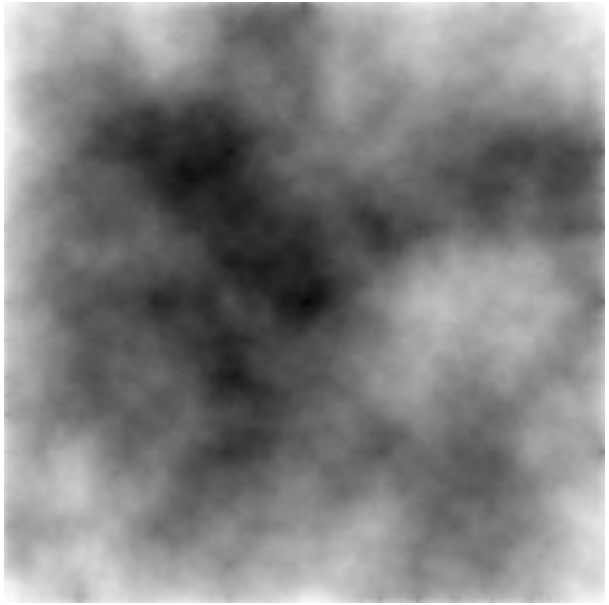
Depunere de particule

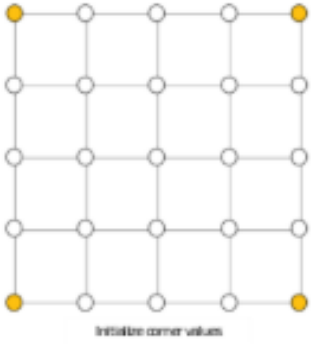
- Adăugăm o unitate de înălțime unei celule aleasă aleator.
- Dacă o celulă vecină se află la mai mult de două unități distanță, unitatea de înălțime adăugată va cădea pe celula vecină.

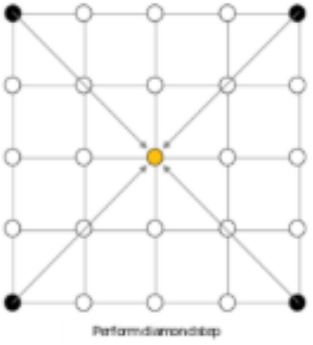


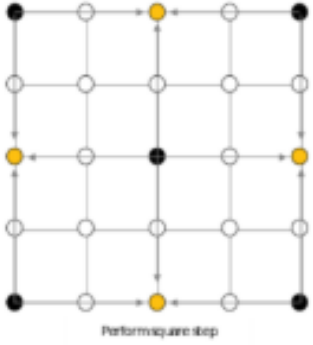
Algoritmul Diamant-Pătrat

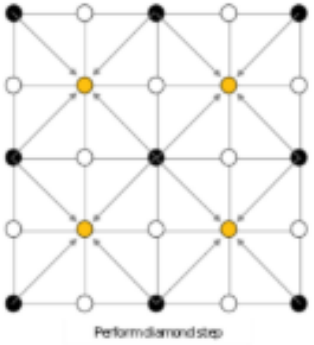
- Se folosește de o hartă de înălțimi cu rezoluție redusă.
- Produce hărți de înălțime cu aspect natural și divers.
- Inițial se atribuite valori de înălțime în vârfurile hărții.
- Ulterior, se aplică repetitive pașii diamant și de pătrat și se face media vârfurilor.

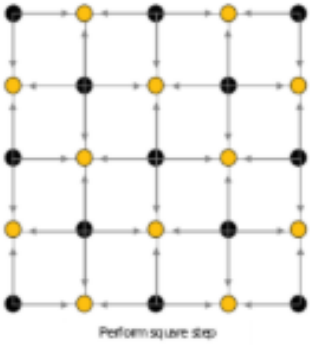






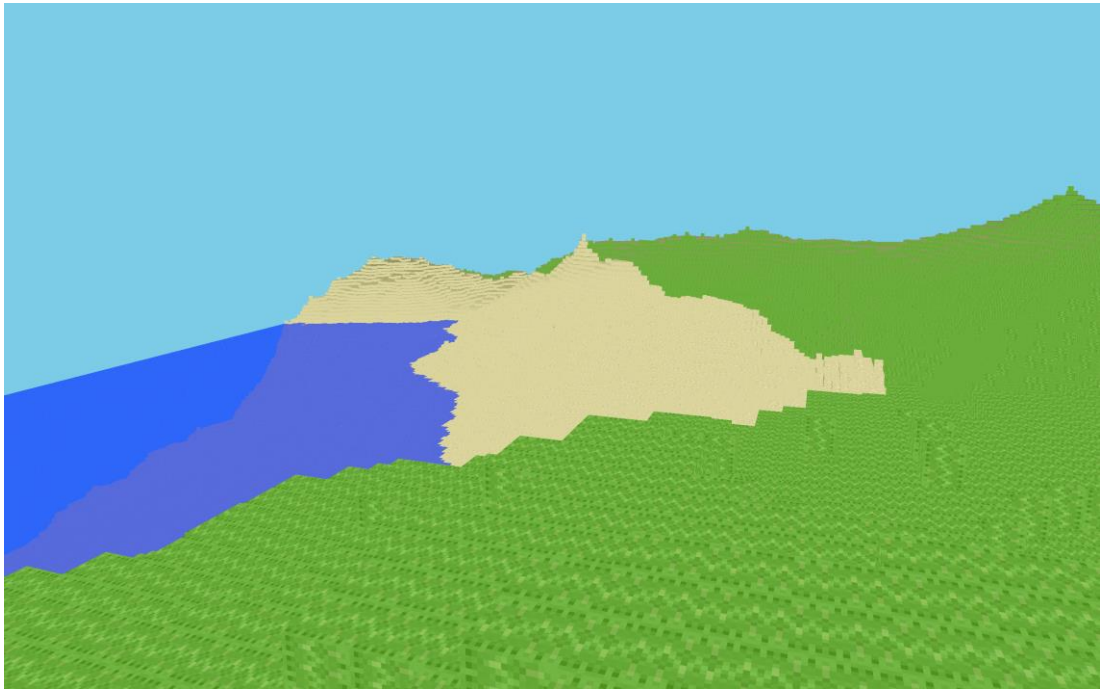






Algoritmul Diamant-Pătrat

https://medium.com/@nickobrien/diamond-square-algorithm-explanation-and-c-implementation-5efa891e486f



Teren folosind algoritmul Diamant-Pătrat

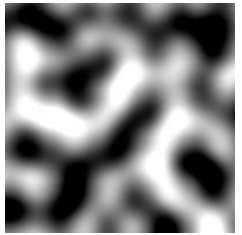
https://medium.com/@nickobrien/diamond-square-algorithm-explanation-and-c-implementation-5efa891e486f

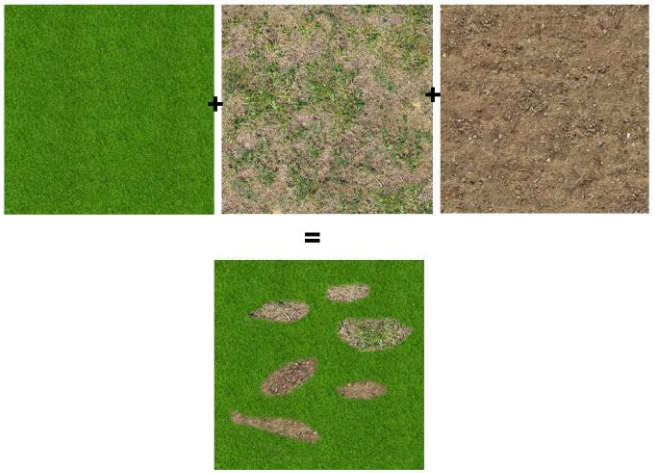
Algorithm 1: Diamond-square

Data: mapSize (power of 2), $\alpha \in \mathbb{R}^+$
Main ()
  map ← *blank heightmap of size* (mapSize + 1, mapSize + 1)
  initialize corners of the map
  size ← mapSize
  while size > 1 **do**
    diamondStep(size)
    squareStep(size)
    size ← size / 2
  end
  return map
Procedure `diamondStep()`
  for x in 0, size ... (mapSize − size) **do**
    for y in 0, size ... (mapSize − size) **do**
      corners ← map[(x, y), (x + size, y), (x, y + size), (x + size, y + size)]
      map[x + $\frac{\text{size}}{2}$, y + $\frac{\text{size}}{2}$] ← mean(corners) + $\alpha \cdot$ size $\cdot$ uniform(-1, 1)
    end
  end
Procedure `squareStep()`
  for x in 0, size ... mapSize **do**
    for y in 0, size ... mapSize **do**
      coordsA ← [(x, y), (x + size, y), (x + $\frac{\text{size}}{2}$, y + $\frac{\text{size}}{2}$), (x + $\frac{\text{size}}{2}$), (y − $\frac{\text{size}}{2}$)]
      coordsB ← [(x, y), (x, y + size), (x + $\frac{\text{size}}{2}$, y + $\frac{\text{size}}{2}$), (x − $\frac{\text{size}}{2}$), (y + $\frac{\text{size}}{2}$)]
      midPointsA ← *values of map at all coordsA which lie in map*
      midPointsB ← *values of map of all coordsB which lie in map*
      map[x + $\frac{\text{size}}{2}$, y] ← mean(midPointsA) + $\alpha \cdot$ size $\cdot$ uniform(-1, 1)
      map[x, y + $\frac{\text{size}}{2}$] ← mean(midPointsB) + $\alpha \cdot$ size $\cdot$ uniform(-1, 1)
    end
  end

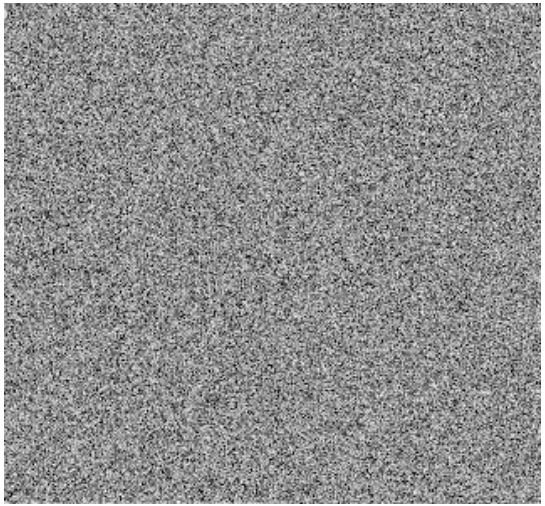
Funcții de Zgomot

- Zgomotul este un semnal aleator, o funcție utilizată pentru a genera valori aleatorii (PRNG).
- Utilizate pentru a adăuga varietate și elemente aleatoare în imagini, sunete, textura, terenuri etc.
- Cel mai simplu tip de zgomot este Zgomotul Alb (White Noise).
- Se utilizează alături de funcții de interpolare pentru un aspect neted.

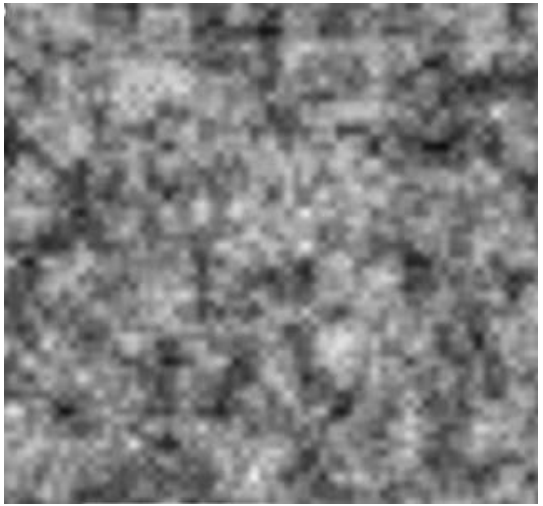




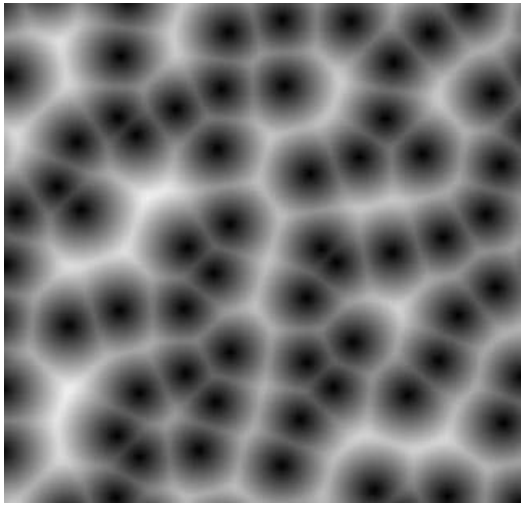
Funcții de zgomot



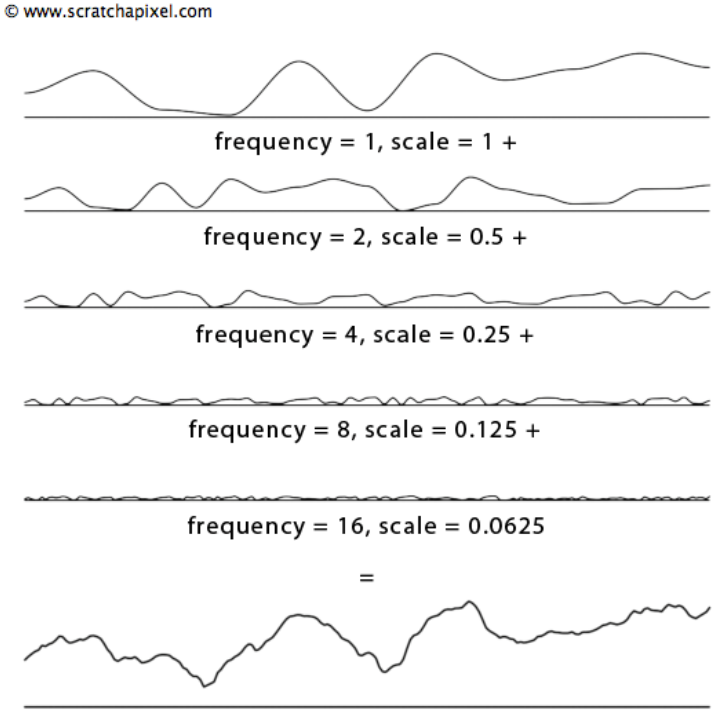
White Noise



Value Noise



Voronoi Noise

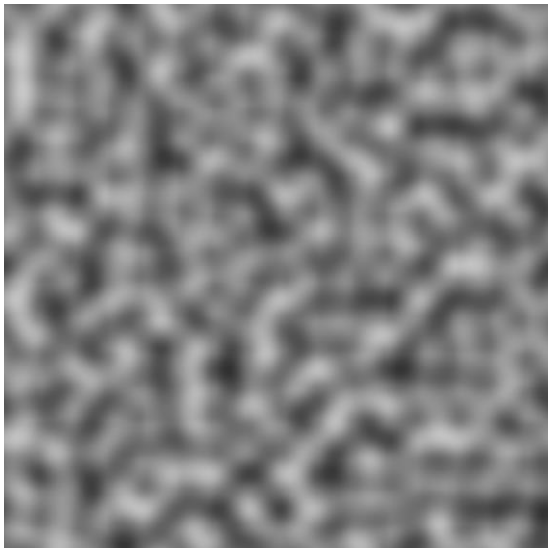


Efectul octavelor în generarea de zgomot

https://www.scratchapixel.com/lessons/procedural-generation-virtual-worlds/procedural-patterns-noise-part-1/simple-pattern-examples.html

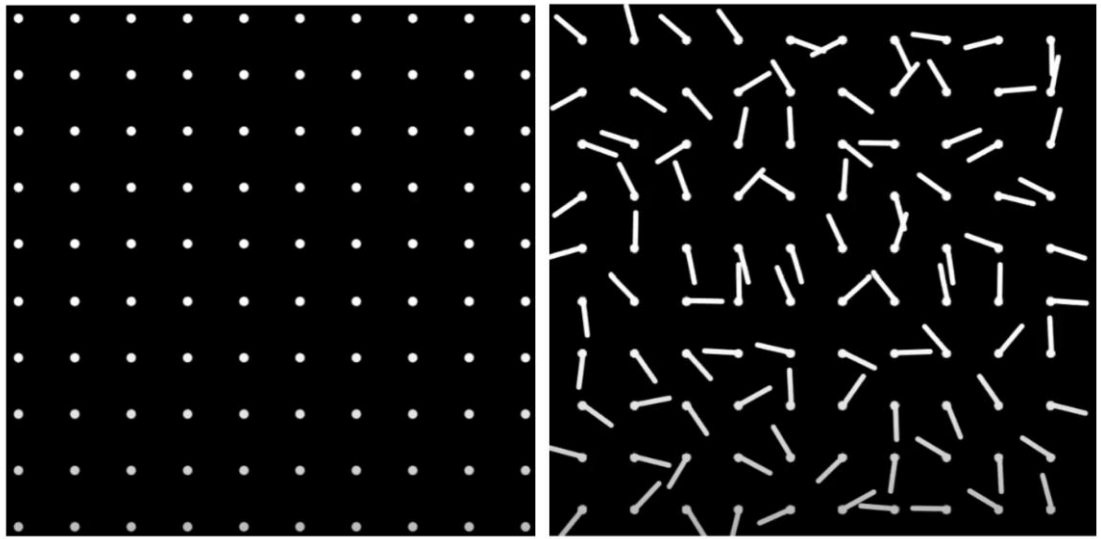
Zgomotul Perlin

- Dezvoltat de Ken Perlin în 1983.
- Printre **cele mai utilizare funcții de zgomot**.
- Produce un aspect natural, ideal pentru texturare, aplicarea unei variabile pseude-aleatoare și generare de teren cu treceri naturale între zonele înalte și joase.
- Funcție de zgomot bazată pe gradientul pe latici.
- Utilizată în generarea de teren în jocuri precum Terraria sau Minecraft.



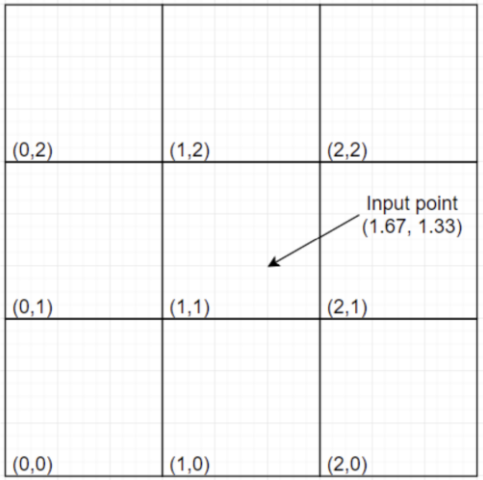
Zgomotul Perlin

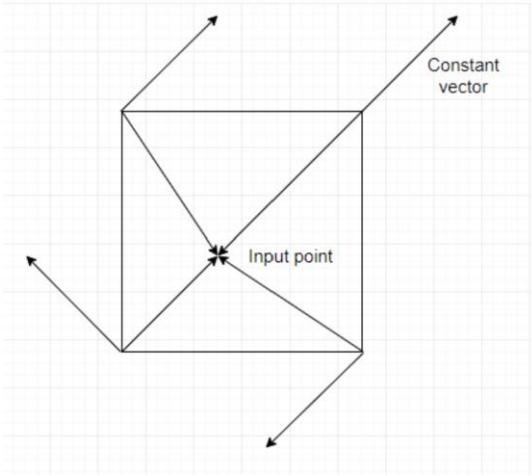
- Utilizează o matrice de latici, fiecare având un vector gradient ales aleator in intervalul [-1, 1].



Zgomotul Perlin

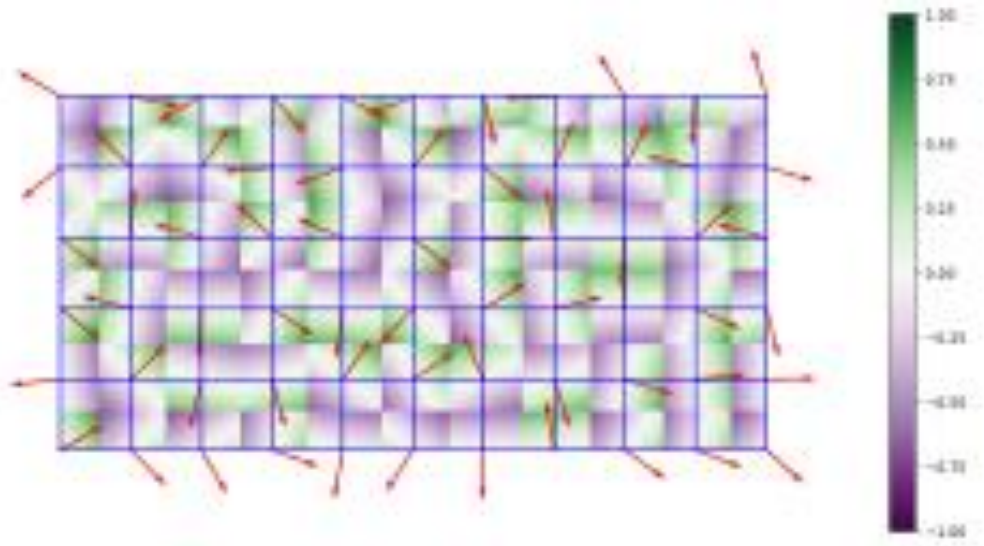
- Pentru fiecare punct în care se calculează funcția de zgomot se identifică celula în care se află. Apoi se calculează vectorii de offset de la vârfurile celulei până la punctul din celulă.





Zgomotul Perlin

- Se calculează produsul scalar între vectorul gradient și vectorul de offset, iar mai apoi se aplică o funcție de interpolare între vârfuri.



Resurse

N. Shaker, J. Togelius, and M. J. Nelson, *Procedural content generation in games*. Springer, 2016.

T. X. Short and T. Adams, *Procedural generation in game design*. A K PETERS, 2017.

K. Perlin, “An image synthesizer,” ACM SIGGRAPH Computer Graphics, vol. 19, no. 3, pp. 287–296, Jul. 1985, doi: 10.1145/325165.325247.

Jfokus, “Reinventing Minecraft world generation by Henrik Kniberg,” YouTube. May 9,202. [Online]. Available: https://www.youtube.com/watch?v=ob3VwY4JyzE

“Procedural Content Generation in Computer Games (Summer 2021/22) – Gamedev.Cuni.Cz.” https://gamedev.cuni.cz/study/courses-history/courses-2020-2021/procedural-content-generation-in-computer-games-summer-2021-22/

Curs - „Limbaje Formale și Automate” –UB, FMI, Informatică An 1