Perlin-zaj

Pintér Bálint

January 24, 2026

Tartalomjegyzék

1 Perlin-zaj

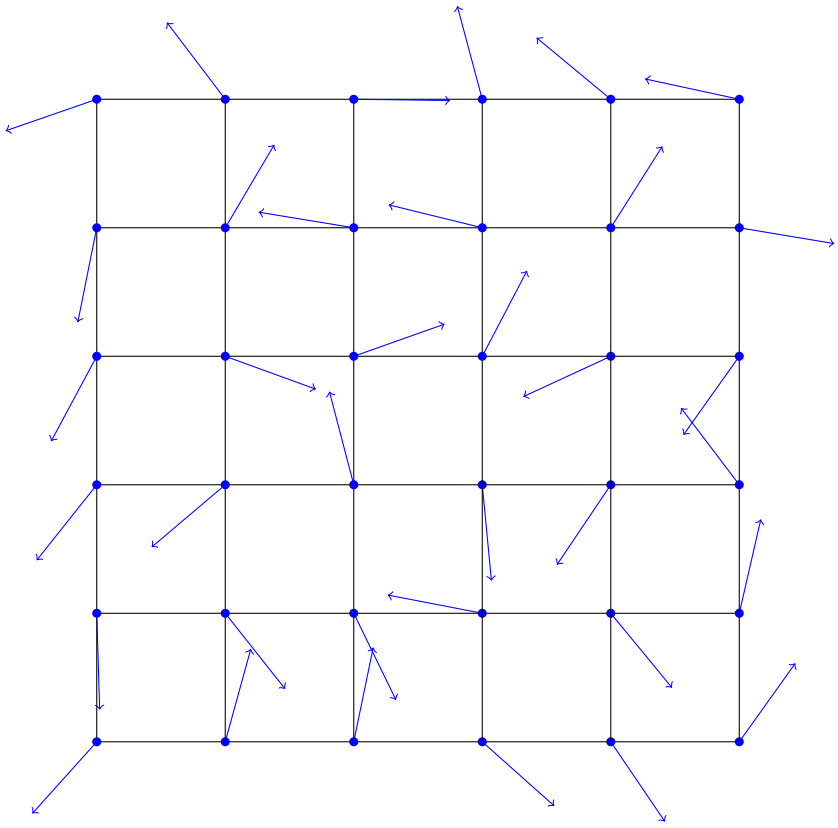
A Perlin-zaj egy zajgeneráló algoritmus, amely véletlenszerű, de összefüggő zajt generál. Így a természetben előforduló véletlenszerű jelenségeket jól lehet vele szimulálni, mint például domborzatok, felhők vagy a víz hullámzása. Tetszőleges n dimenzióra létrehozható, de jellemzően az 1-től a 4. dimenzióig alkalmazzák. A kódban egy kétdimenziós Perlin-zaj van implementálva.

2 Előkészítés

A Perlin-zaj hatékony generálásához két adat inicializálására van szükség: egy gradiens táblára és egy permutációs táblára.

2.1 Gradiens tábla

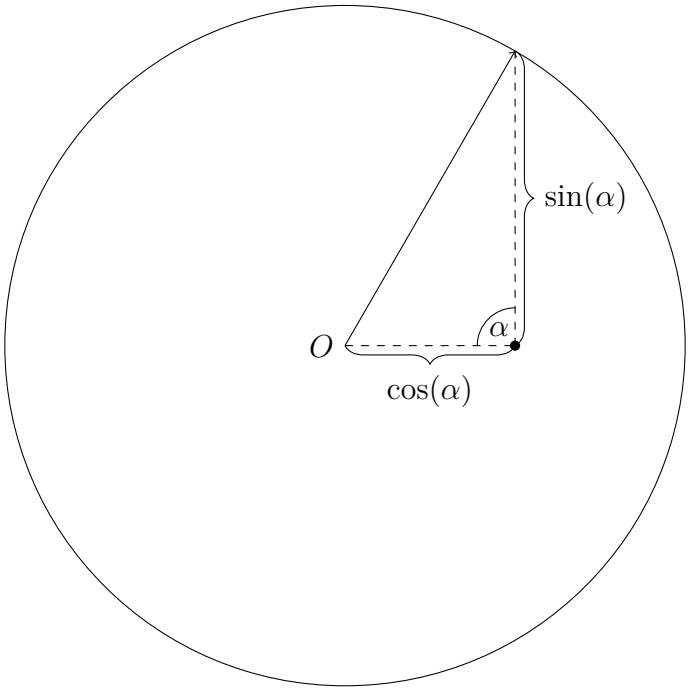
A Perlin-zaj egy úgynevezett gradiens-zaj. Eszerint rácspontokat határozunk meg, amikhez egy véletlenszerű vektort rendelünk. A gradiens tábla ezeket a véletlenszerű vektorokat tárolja. A vektorok dimenziószáma megegyezik a zaj dimenziószámával. (Kétdimenziós zaj $\rightarrow$ kétdimenziós vektor)



1. Ábra:
A záj rácsainák szemléltetése.

2.1.1 Vektor generálás

Generálunk egy véletlenszerű számot $[0; 2\pi[$ intervallumban. Majd egyszerű trigonometriával a szöget egy vektorrá alakítjuk, ahol a vektor x komponense a véletlen szög koszinusza, és az y komponense a szög szinusza.



2. Ábra:
A vektorok előállításának szemléltetése.

2.2 Permutációs tábla

A permutációs tábla kezdetben 0-tól 255-ig tartalmazza a számokat növekvő sorrendben. Ezt a listát egy véletlenszám-generátor segítségével összekeverjük és önmaga után fűzzük (ezzel egy 512 elemű tömböt kapunk). Így a hashelésnél elkerülhető a túlindexelés, ami gyorsítja a zajgenerálást, mivel elhagyható a túlindexelésre való ellenőrzés.

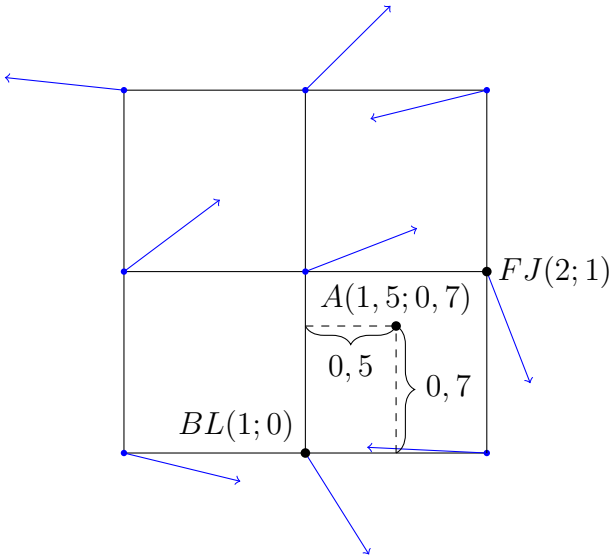
1. Algoritmus: Permutációs tábla létrehozása

```
   Konstans: MaxP=512
   Típus: VéletlenSzámGenerátor=Osztály (
          jelenlegiSzám:Egész
          Függvény Következő:Egész
   )
1  Eljárás PermutaciosTablaGeneral (Változó: PermutaciosTabla:Tömb(1..MaxP:Egész),
2  Rand:VéletlenSzámGenerátor ):
       Változó: i, j, temp:Egész
3
4      Ciklus i := 1-től 256-ig
5          PermutaciosTabla[i] := i
6      Ciklus vége
7
8      Ciklus i := 256-tól 2-ig −1-esével
9          j := Rand.Következő() Mod (i + 1)
10         temp := PermutaciosTabla[i]
11         PermutaciosTabla[i] := PermutaciosTabla[j]
12         PermutaciosTabla[j] := temp
13     Ciklus vége
14
15     Ciklus i := 1-től 256-ig
16         PermutaciosTabla[i + 256] := PermutaciosTabla[i]
17     Ciklus vége
18 Eljárás vége
```

3 Zajszámítás

3.1 Rácspontok meghatározása

Először meghatározzuk, hogy az adott (x, y) pont melyik négyzetbe tartozik ezt a bitenkénti ÉS 255 művelettel tesszük, így az eredmény a $[0; 255]$ tartományba fog esni: ha az érték nagyobb 255-től, akkor visszafordul az intervallum elejére (pl. 256-ból 0 lesz). Ezt elvégezve az x-re és y-ra megkapjuk a bal alsó rácspont koordinátáit. A bal alsó rácspont koordinátáihoz hozzáadva egyet majd egy bitenkénti ÉS 255 művelettel megkapjuk a jobb felső rácspont koordinátáit. A négyzeten belüli pontot úgy kapjuk meg, hogy a szám egész részét elhagyjuk.



3. Ábra:
A rácspont koordinátáainak szemléltetése.

Pszeudókódban megvalósítva:

2. Algoritmus: Rácspontok és négyzeten belüli koordináták kiszámolása

```
Típus: Rácspont=Strukt (
        balAlsóPontX, balAlsóPontY:Egész
        jobbFelsőPontX, jobbFelsőPontY:Egész
        relatívX, relatívY :Valós
)
1  Függvény RacspontKiszamolasa(Konstans: x, y: Valós) : Rácspont:
       Változó: jelenlegiRácspont: Rácspont
2      jelenlegiRácspont.balAlsóPontX := (Egész)floor(x) & 256 + 1
3      jelenlegiRácspont.balAlsóPontY := (Egész)floor(y) & 256 + 1
4
5      jelenlegiRácspont.jobbAlsóPontX := (jelenlegiRácspont.balFelsőPontX + 1) & 256
6      jelenlegiRácspont.jobbAlsóPontY := (jelenlegiRácspont.balFelsőPontY + 1) & 256
7
8      jelenlegiRácspont.relatívX := x− floor(x)
9      jelenlegiRácspont.relatívY := y− floor(y)
10     RacspontKiszamolasa := jelenlegiRácspont
11 Függvény vége
```