Perlin-zaj

Pintér Bálint

2026. január 24.

Tartalomjegyzék

1. Perlin-zaj

A Perlin-zaj egy gradiens alapú zajgenerálási algoritmus, amelynek a célja a véletlenszerű, de összefüggő zaj létrehozása. Segítségével a természetben előforduló véletlenszerű jelenségeket jól lehet 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.

1.1. Működése összefoglalva

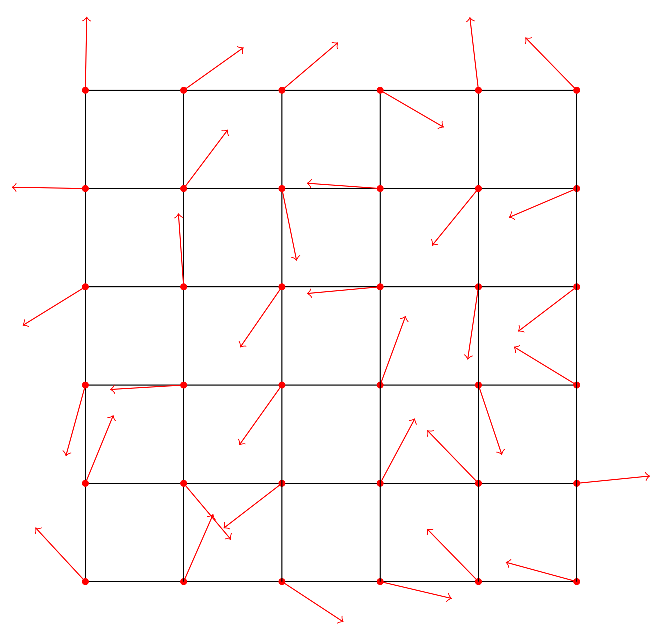
1. **Rács meghatározása:** A zaj dimenziójában egy szabályos rácsot határozunk meg, amelynek minden sarkához egy véletlenszerűen generált egységvektort rendelünk. A rácsvonalak jellemzően az egész koordinátáknál helyezkednek el.
2. **Rácsnégyzeten belüli vektorok kiszámítása:** Kiszámoljuk a rácsnégyzetén belüli pontba a rács sarkaiból mutató vektorokat.
3. **Skaláris szorzás:** Az adott sarokból a pontba mutató vektornak és az adott sarokhoz rendelt vektornak vesszük a skaláris szorzatát.
4. **Interpoláció:** A kapott skaláris szorzatokat végül tengelyekként interpoláljuk egy simító függvénnyel. Például a kétdimenziós zajnál először az x tengely mentén interpolálunk majd a kapott részeredményeket az y tengely mentén interpoláljuk.

2. Előkészítés

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

2.1. Gradiens tábla

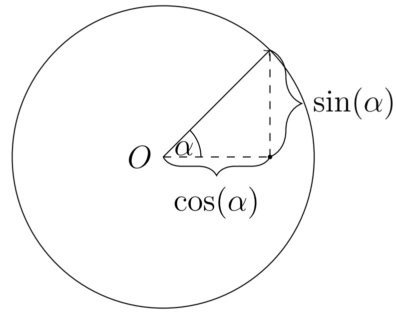
A Perlin-zaj egy úgynevezett gradiens-zaj. Eszerint rácspontokat határozunk meg, amikhez egy véletlenszerű vektort rendelünk. A gradiens tábla ezeket a véletlenszerű vektorokat tárolja. A vektorok dimenziószáma megegyezik a zaj dimenziószámával. (Kétdimenziós zaj $\rightarrow$ kétdimenziós vektor)



1. Ábra
A záj rácsainak 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.

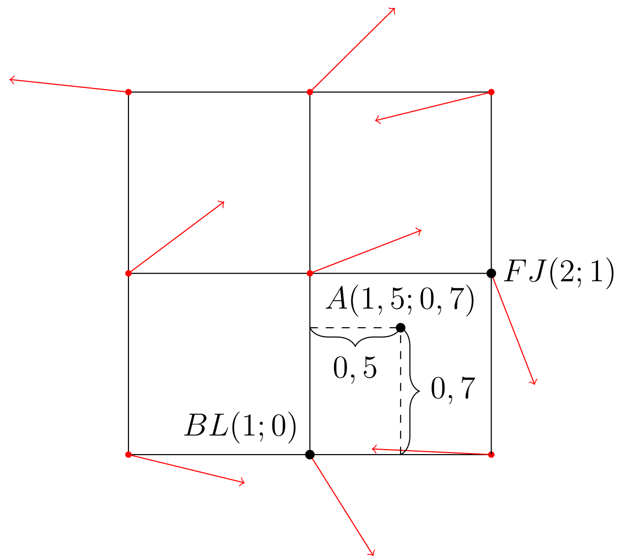
1. Algoritmus: Permutációs tábla létrehozása

```
Konstans: MaxP=512
Típus: VéletlenSzámGenerátor=Osztály (
    jelenlegiSzám:Egész
    Függvény Következő:Egész
)
Eljárás PermutaciosTablaGeneral (Változó: PermutaciosTabla:Tömb(1..MaxP:Egész),
Rand:VéletlenSzámGenerátor ):
    Változó: i, j, temp:Egész

    Ciklus i := 1-től 256-ig
        PermutaciosTabla[i] := i
    Ciklus vége

    Ciklus i := 256-tól 2-ig −1-esével
        j := Rand.Következő() Mod (i + 1)
        temp := PermutaciosTabla[i]
        PermutaciosTabla[i] := PermutaciosTabla[j]
        PermutaciosTabla[j] := temp
    Ciklus vége

    Ciklus i := 1-től 256-ig
        PermutaciosTabla[i + 256] := PermutaciosTabla[i]
    Ciklus vége
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=Rekord (
        balAlsóPontX, balAlsóPontY:Egész
        jobbFelsőPontX, jobbFelsőPontY:Egész
        relatívX, relatívY :Valós
)
Függvény RacspontKiszamolasa(Konstans: x, y: Valós) : Rácspont:
    Változó: jelenlegiRácspont: Rácspont
    jelenlegiRácspont.balAlsóPontX := (Egész)floor(x) & 256 + 1
    jelenlegiRácspont.balAlsóPontY := (Egész)floor(y) & 256 + 1

    jelenlegiRácspont.jobbAlsóPontX := (jelenlegiRácspont.balFelsőPontX + 1) & 256
    jelenlegiRácspont.jobbAlsóPontY := (jelenlegiRácspont.balFelsőPontY + 1) & 256

    jelenlegiRácspont.relatívX := x− floor(x)
    jelenlegiRácspont.relatívY := y− floor(y)
    RacspontKiszamolasa := jelenlegiRácspont
Függvény vége
```

3.2. Gradiens vektorok kiválasztása

A gradiens vektorokat a permutációs tábla segítségével válasszuk ki. Vesszük a permutációs tábla x-edik elemét, hozzáadjuk az y értékét, majd az így kapott összeget használjuk indexként a permutációs táblában. Az így kapott eredmény lesz az indexe a gradiens vektornak a gradiens táblából.

3. Algoritmus: Gradiens vektor kiválasztása

```
Típus: Vektor=Rekord (
    x:Egész
    y:Egész
)
Konstans: MaxP=512, MaxG=256
Változó: PermutaciosTabla:Tömb(1..MaxP:Egész)
Változó: GradiensTabla:Tömb(1..MaxG:Egész)

Függvény Hash(Konstans: x, y: Egész) : Egész:
|   Hash := PermutaciosTabla[PermutaciosTabla[x] + y]
Függvény vége

Függvény GradiensVektorKivalaszt(Konstans: x, y: Egész) : Vektor:
|   GradiensVektorKivalaszt := GradiensTabla[hash(x, y)]
Függvény vége
```