Obecný vzdáleností řízeny dithering

Na základě potřeby aplikace binární gradace na nerovnoměrné 3D mřížce jsem upravil známý princip ditheringu na obecný princip který není vázaný na rovnoměrnou 2D čtvercovou mřížku, ale zároveň si zachoval jeho vlastnosti.



Dithering tak jak je definován doposud slouží k náhodnému rozložení kvantové chyby[[1]](#endnote-1) vznikající při snížení rozsahu nabývaných hodnot, tak aby se předešlo ztrátě plynulosti v makro měřítku.

# Náhodný dithering:

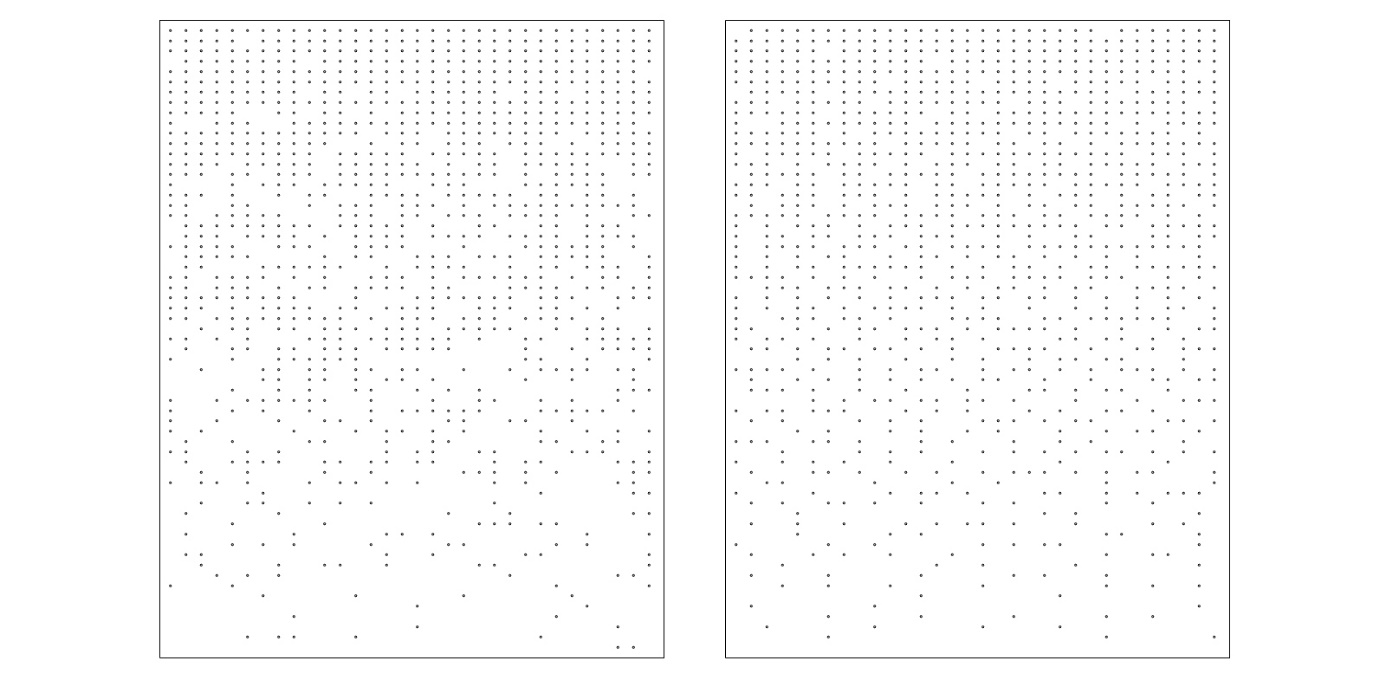
Náhodný dithering využívá porovnávaní náhodně vybraného čísla a původní hodnoty ke stanovení nové hodnoty. Následující algorytmus je platný pro binární dithering, kde knoečná množina nabývaných hodnot nabýbvá pouze minima či maxima z povodní množiny.

## Algorytmus:

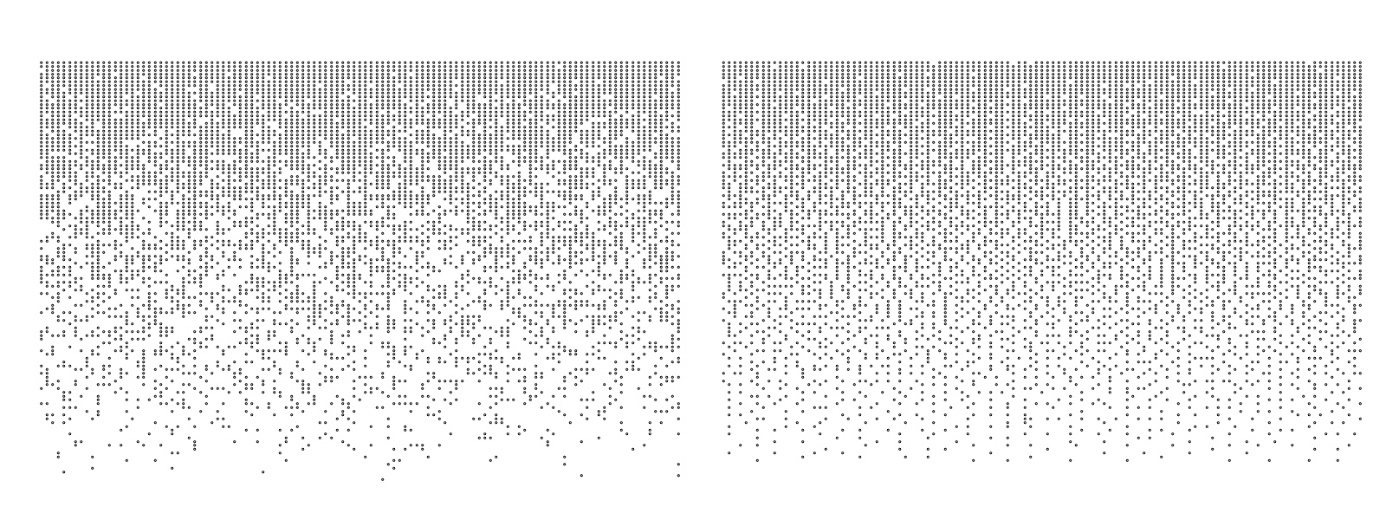
1. Pro každý z množiny bodů je přiřazeno náhodné číslo *r* v rozmezí mezi minimální a maximální hodnotou.
2. Pokud je je výsledná hodnota v opačném případě je

Jak za algorytmu vyplývá každý z bodů je zpracováván samostatně. To přináší řadu výhod a nevýhod. Hlavní výhodou je, že algorytmus funguje i na náhodně rozmístěných bodech, či ve více než dvou-dimenzionálním rozložení. Nevýhodou je potom to, že body bez vztahu k okolí mají tendenci často tvořit tzv. ostrovy a jezera.

# Můj obecný dithering (distance-based dithering)



Vlevo: Náhodný dithering, vpravo můj obecný dithering



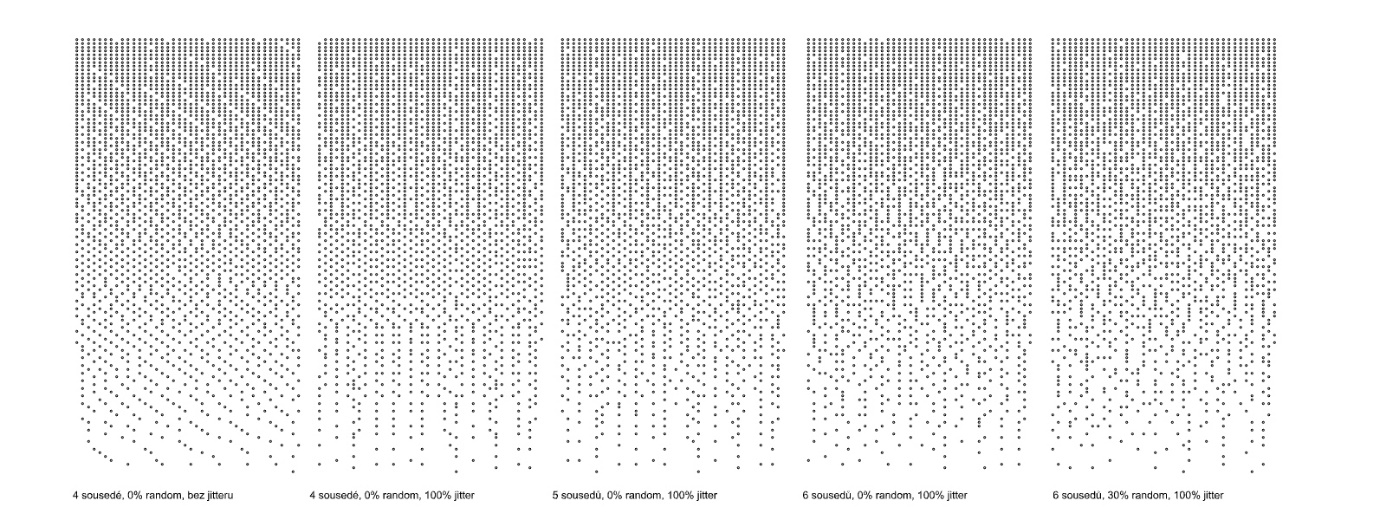
Vlevo: Náhodný dithering, vpravo můj obecný dithering

Aby se přdešlo výskytu „jezer“ a „ostrovů“ je třeba zpracovávat body ve vztahu ke svému okolí. Proto jsem navrhl tento algorytmus:

Algorytmus:

1. Narozdíl od regulárního 2D pole nelze snadno jednotlivé body setřídit do matice, proto je lepší je náhodně seřadit do pole, bez vazby na jejich vzájemnou polohu.
2. Následně je pole bodů postupně procházeno a je nalezena nejbližžší možná hodnota v nové doméně. Například budeme li odstíny šedi převádět na Černou a bílou bude ke 40% šedé blíže bílá než černa, proto bude vybrána bílá jako nová hodnota.
3. Určí se vzniklá odchylka *e* rovna rozdílu nové od původní hodnoty.
4. Pro zpracovávaný bod je nalezeno *n* nejbližších sousedních bodů. Počet *n* významně ovlivňuje výslednou podobu ditheringu. Obecně platí, že nejelších výsledků dosahují hodnoty mezi 5 až 9 .
5. U každého z n nejbližších bodů je zkontrolováno, zda byl již dříve zpracován. Takové body (stejně jako v případě standartního ditheringu) jsou vyřazeny z dalšího zpracování.
6. Každému z *m* nejbližších nezpracovaných bodů je přidělena váha *r* na základě jeho vzdálenosti *d* od zpracovávaného bodu.
7. Každému z *m* bodů je následně přičtena část vzniklé odchylky *e* podle váhy vypočtené v předhozím kroku. Nová hodnota bodů je určena takto:  
     
   Tímto způsobem je odchylka vzniklá v kroku č. 3 rozdělena mezi sousední nezpracované body. Pouze v případě, že všech *n* nejbližších bodů bylo zpracováno je odchylka vyhozena.
8. Právě zpracovaný bod je označen jako zpracovaný a jeho hodnota se následně již nemění. Celý výpočet pokračuje s dalším bodem v poli bodů.

Tímto způsobem lze dosáhnout velice rovnoměrného rozložení hodnot v okolí. Na druhou stranu toto řešení má náchylnost ke tvorbě liniových a šachovnicových artefaktů. To lze poměrně dobře odstranit náhodným pozměněním vstupních hodnot. Dochází tím způsobem k jemnému znehodnocení kvality, ale výsledky jsou mnohem náhodně vypadající.



*Výsledky s různými parametry*

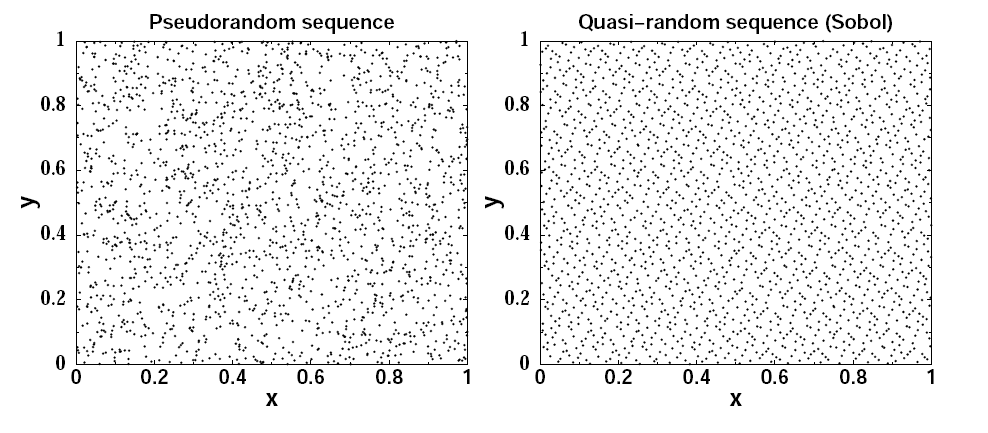
# Webové rozhraní

Po dilematu z minulého semestru jsem se rozhodl pro využití systému „infinite scroll“ nabízející mnohem mnohem rychleji větší množství možných mutací a tím i větší zpracovanou populaci pro další výpočet.

Zároveň jsem se rozhodnul použít pouze pozitivní hodnocení pro zjenodušení celé uživatelkého rozhraní i samotného algorytmu.

## Krok 0:

Na začátku je vygenerováno určité množství bodů v n-rozměrném prostoru, kde n je rovno počtu genů. Rozložení bodů je buďto náhodné nebo lépe kvazi-náhodné (například Sobol sequence, Hammersley sequence) viz: <https://en.wikipedia.org/wiki/Low-discrepancy_sequence>, či <https://ac.els-cdn.com/0022314X8890025X/1-s2.0-0022314X8890025X-main.pdf?_tid=2657588f-ce64-46ec-99d2-27644853f77f&acdnat=1550515249_db49a72c1284734a603717fa9d11871e> .



*Srovnání náhodného a kvazi náhodného rozložení*

|  |  |
| --- | --- |
| **Náhodné rozložení** | **Kvazi náhodné rozložení** |
| + Jednoduchá implementace  + výpočetní náročnost  + lepší pravděpodobnostní rozložení v prohledávaném prostoru (úměrné kvalitě generátoru náhodných čísel) | + Lepší pokrytí prohledávaného prostoru |
| - Nižší pestrost nabízených výsledků | - Větší riziko syntaktické chyby  - Složitá kontrola funkčnosti ve více jak trojrozměrném prostoru. |

Srovnání: <http://extremelearning.com.au/a-simple-method-to-construct-isotropic-quasirandom-blue-noise-point-sequences/>

## Generace 1:

Ve chvíli kdy je vybrán jeden preferovaný kandidát by měly evolučními algoritmy vznikat nové body tak aby nabízeli stále více bodů v okolí preferovaného bodu.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Náhodný návrh** | **Nově Preferovaný návrh** | **Starší preferovaný návrh** |
| Binární konstantní Fitness |  |  |  |
| časem klesající fitness |  |  | Age = počet novějších preferencí + 1  min = 0.25 (minimální fitness) |

1. https://en.wikipedia.org/wiki/Dither [↑](#endnote-ref-1)