Babeş – Bolyai Tudományegyetem

Matematika és Informatika Kar

Cluj-Napoca, Romania

# Valós idejű procedurális domborzatmodellezés Perlin zajjal

**Dr. Szenkovits Ferenc**

egyetemi docens
Babeş-Bolyai Tudományegyetem
Matematikai és Informatikai Kar
Mechanikai és Csillagászati Tanszék

**Dr. Soos Anna**

adjunktus
Babeş-Bolyai Tudományegyetem
Matematikai és Informatikai Kar
Numerikus Analízis és Statisztika Tanszék

**Osváth-Boros Róbert**

Babeş-Bolyai Tudományegyetem
Matematikai és Informatikai Kar
Informatika szak, IV. évfolyam

*IX. Erdélyi Tudományos Diákköri Konferencia - Kolozsvár, 2006. november 25-26*

## Absztrakt

A domborzat modellezés a virtuális valóság egyik fontos alkotóeleme. A procedurális módszerek a természetben előforduló „végtelenség” érzetét szintetizálják, minimális tárolási igénnyel. Ezek a módszerek már a 80-as években megjelentek, de valós idejű implementálásuk még most is kezdetleges. Ahhoz hogy egy virtuális valóság hiteles legyen, a procedurális domborzatgeneráló technikáknak valós időben kell működniük. A „hardware” korlátok miatt ki kell dolgozni egy olyan módszert ami hozzáférhetővé tegye a procedurális domborzatkialakítást az egyszerű PC környezetekhez.

## 1. Bevezetés

Domborzatfelszínt számítógépen létrehozni többféleképpen lehet, ezek közül az egyik legsikeresebb a fraktál domborzatfelszín modell. A Perlin zajjal generált domborzatmodellek, tulajdonképpen a fraktál domborzatok Perlin-zaj alapú ága. Ahhoz hogy fraktál domborzatmodellt létrehozzunk, 4 fontos elemre van szükség: egy alapfüggvényre, ami a domborzat alakját megadja (pl. Perlin-alap), a fraktál dimenziójára (az amplitúdó módosulása minden iterációban), az oktávok (iterációk) számára, illetve a frekvencia módosulási tényezőjére. [1.1] (T&MPA - Building Random Fractals - F. Kenton Musgrave)

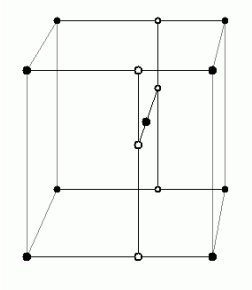
### 1.1 Perlin zajfüggvénye

Perlin zajfüggvénye $R^n$-en értelmezett, egész számokban csomópontokat képző rácshoz igazított pseudo-random spline függvény, ami a véletlenszerűség hatását kelti, de ugyanakkor rendelkezik azzal a tulajdonsággal, hogy azonos bemeneti értékekre, azonos a függvényértéket térit vissza.

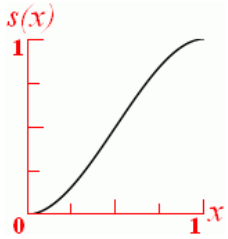
$$f : \mathbb{R}^n \rightarrow [-1, +1]$$

A gyakrabban használt *n* értékei az n=1, n=2 illetve n=3. (1-animáció, 2-egyszerű textúrák, 3-bonyolultabb textúrák).

Perlin-zaj generálására az eljárás a következő: adva van egy bemeneti pont. Minden környező rács-csomópontra választani kell egy pseudo-random értéket egy előre generált halmazból (mivel a csomópontok koordinátái egész számok, ezeket használjuk az eredmény kiválasztására). Majd interpolálni kell az így megkapott csomópontokhoz rendelt értékek között, valamilyen S görbét használva. (pl. $3t^2-2t^3$). [1.2] (Kent-Perlin: The noise machine)



(1. ábra: a 8 környező csomópont, illetve a 7 interpoláció során meghatározott végső érték)

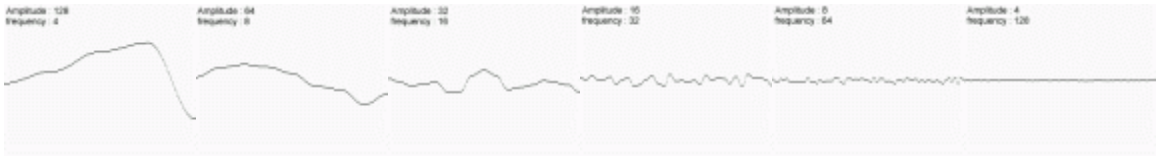



(2. ábra: az interpoláció értékét meghatározó S-görbe: $f(t)=3t^2-2t^3$)

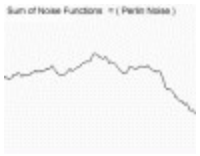
## 1.2 Perlin alapú fraktál függvények

Perlin zajfüggvényét kifejezésben használva, különböző procedurális mintákat és textúrákat lehet létrehozni. [1.3] (T&MPA – Procedural textures)

Ha ezeket a kifejezéseket fraktál összegben használjuk minden iterációban új adatot vihetünk be a képbe, amik valamilyen módon befolyásolják ezt. Domborzat generálás esetén, az iteráció során a fraktál dimenzióját akarjuk befolyásolni, azaz minden iterációban az amplitudót osztani fogjuk egy bizonyos értékkel. [1.4] (Hugo Elias – Applications of Perlin noise)



(3. ábra: az iteráció lépései: észrevehető hogy az értékek közelednek a 0-hoz [1.4])



(4. ábra: a fraktál összeg [1.4])

## 2. Kísérletek

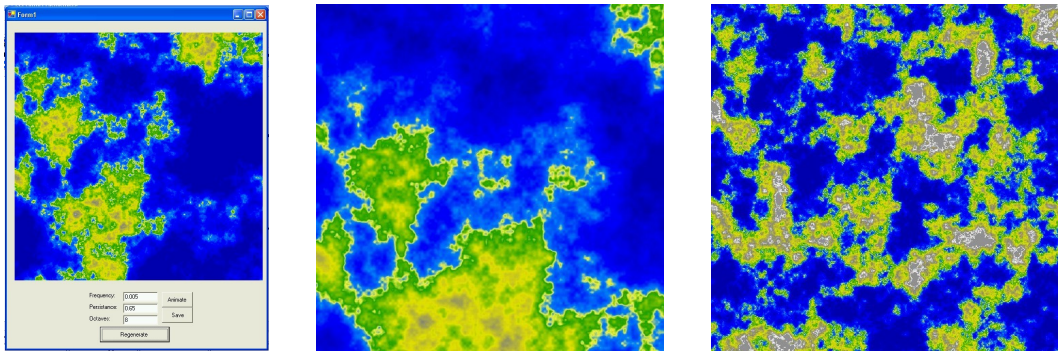
Ahhoz hogy egy Perlin alapú függvény képét szemléltessem, a függvény által meghatározott virtuális domborzatot metszettem egy síkkal, majd abból egy szeletet kiragadva jelenítettem meg a képernyőn egy színskála segítségével.

$$f:[-1,+1]\rightarrow[0,255]$$
$$f(x)=128+x\cdot 128$$

Ha *x* helyére a Perlin függvény által egy adott koordinátán visszatérített értéket teszem, majd egy színskála egészrész f(x)-edik értékét rajzolom a képernyőre.

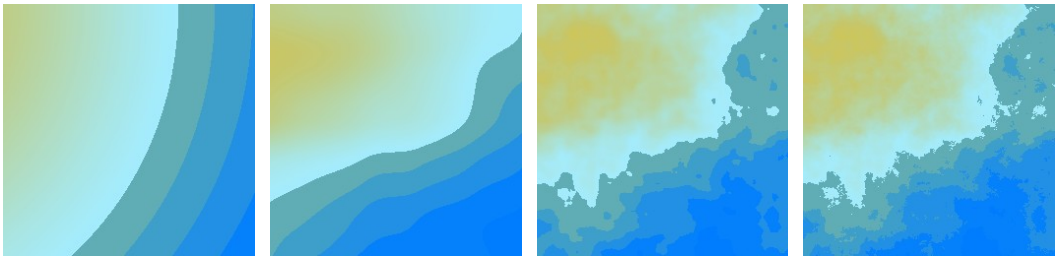
*Algoritmus:*

```
for(i=0;i<256;i++) { for(j=0;j<256;j++) {
        v=perlinNoise(i, j, 0); // a z koordinatat 0-ra allitom
        draw(j, i, colorScale[128 + (int)(v*128)]);
} }
```

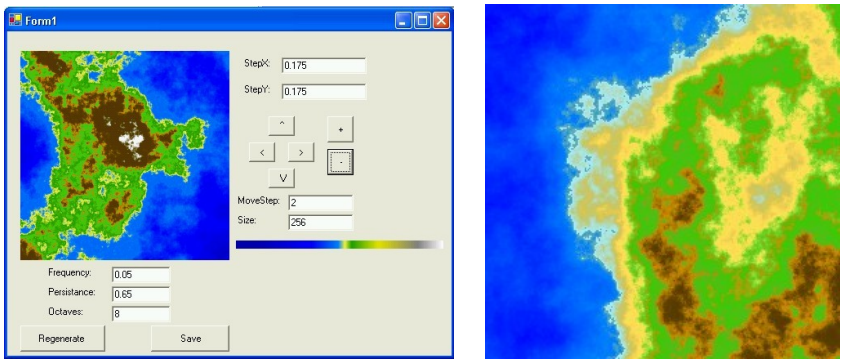


(5. ábra: szeletek a virtuális világból)

További kísérletek során különböző befolyásoló tényezőket vittem be a fraktál függvénybe. Arra törekedtem, hogy a Perlin zaj első oktávja döntse el azt, hogy az adott pont milyen tájegységez tartozik (magasság szerin), majd az iterációs lépésekben, az annak a tájegységnek megfelelő amplitúdó és frekvencia változás paramétereit alkalmazom. (például magas hegy esetén az amplitúdó kis változást kell eredményezzen a fraktálösszegben, míg egy fennsík esetén az amplitúdónak egyből redukálnia kell a részletét a domborzatnak, hogy ezt egy sima felszínre alakítsa)



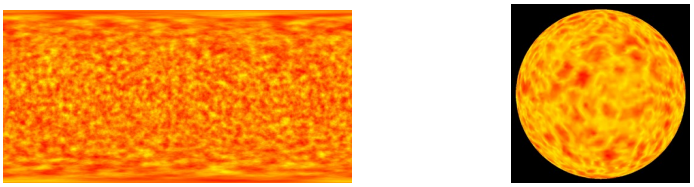
(6. ábra: konstans amplitúdó és frekvencia változás)




(7. ábra: tájegységek elkülönülései: óceán, part-szakasz, alföld, sziklás hegység, dinamikus amplitúdó és frekvencia változás)

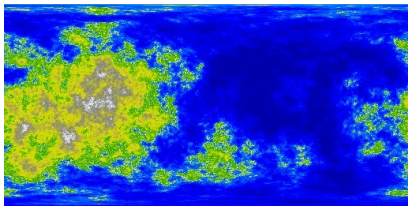
## 3. Eredmények

Ha egy gömb egyenletével, valamint az X és Y értékek segítségével meghatározzuk a Z komponenst, és ezt a zajfüggvénybe helyezzük, akkor a virtuális domborzat egy gömb felszínével való metszetét kapjuk. Majd ezt egy háromszögekből kirakott gömb felszínére rajzoljuk.

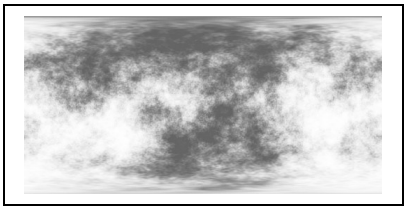


(8. ábra: gömb textúra ráfestve egy gömb felszínére)

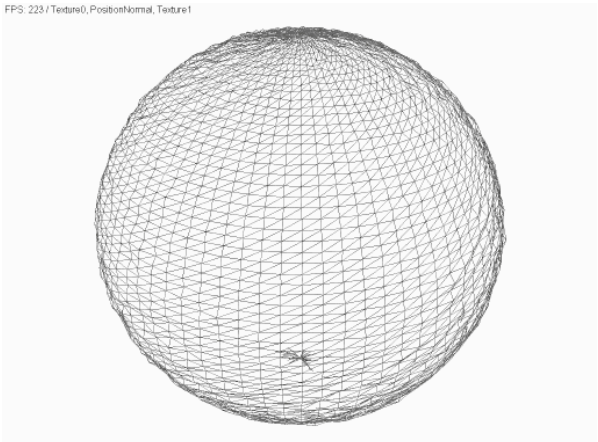
Ezután kiválasztunk két színskálát, amivel 2 különböző textúrát generálunk: domborzatmodell illetve felhőmodell.(mivel a Perlin-zaj természetesnek tűnő textúrákat generál, jól megválasztott színskálával mást is generálni lehet mint domborzatmodelleket)



(9. ábra: első textúra: a domborzat képe)

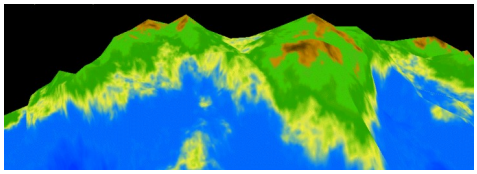


(10. ábra: második textúra: a felhőmodell)



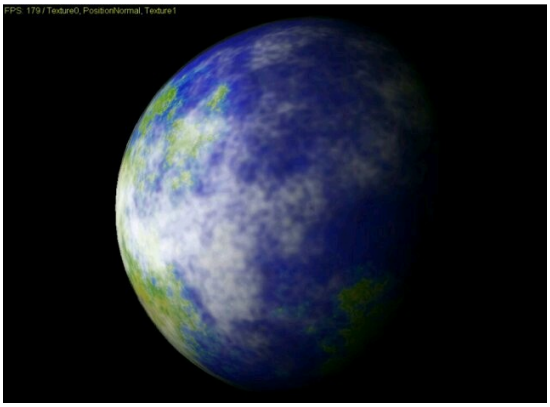

(11. ábra: háromszögekből kirakott gömb, drótháló)

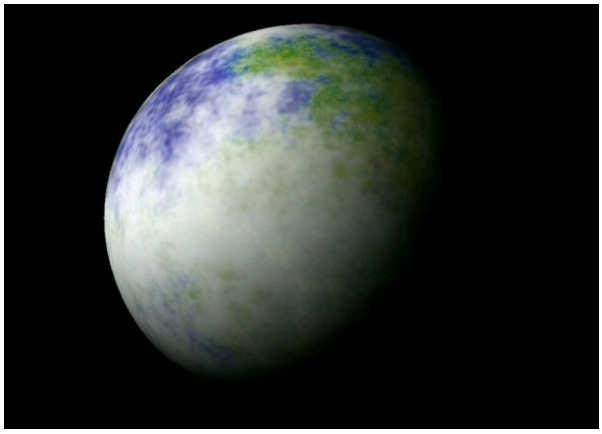
Ha a gömb, háromszög csomópontjain kiszámítjuk a Perlin-zaj értékét, majd a csomópontokat (amik tulajdonképpen vektorok) skalárisan összeszorozzuk a visszatérített értékkel, akkor az így kapott torzítással, domborulatokat idézhetünk elő a gömb felületén)



(12. ábra: az eltorzított gömb „igazi domborzathoz” hasonít)

Végül mindhárom elemet egybedolgozva, a következő procedurális bolygómodellhez jutunk.





## Köszönettel

Dr. Szenkovits Ferencnek – a kutatás kezdeti fázisaiban nyújtott segítségért

Dr. Soos Annanak – a kutatás során illetve a kutatást összefoglaló dolgozat megírásában nyújtott segítségért

## Könyvészet

[1.1] Texturing and Modeling: The procedural aproach – F. Kenton Musgrave: Building Random Fractals

[1.2] Kent Perlin – The Noise Machine (GDC HardCore 1999):
http://www.thenoisemachine.com/talk1/

[1.3] Texturing and Modeling: The procedural aproach – Procedural Textures

[1.4] Hugo Elias – Applications of Perlin Noise
http://freespace.virgin.net/hugo.elias/models/m_perlin.htm

[1.5] Egbert, Musgrave, Peachey, Perlin, Worley - Texturing and Modeling: The procedural aproach 3th edition (2003, Mkf kiadó)

[1.6] libnoise - http://libnoise.sourceforge.net/