Proceduralno generiranje terena s prelomnim algoritmom

Marko Jovčeski

Uvod

Gotovo ste se kdaj ob igranju katere od strateških iger vprašali, na kakšen način se vsakič ustvari nova igralna površina. Kako bi se vi lotili generiranja pokrajine?

Z razvojem računalniške grafike se je porodila potreba po hitrem generiranju vsebin, kar bi poenostavilo dolgotrajnost in zahtevnost ročnega modeliranja. V ta namen se je razvilo proceduralno generiranje vsebin, ki algoritmično ustvarja podatke. Pomemben del tega področja predstavlja proceduralno modeliranje, ki vključuje algoritme za naključno generiranje trirazsežnostnih modelov. Sem spada proceduralno generiranje terena, ki z uporabo algoritmov modelira pokrajine za uporabo v računalniški grafiki in animaciji, filmski industriji in tudi na drugih področjih, kot so geologija, geografija in inženirstvo [1–4].

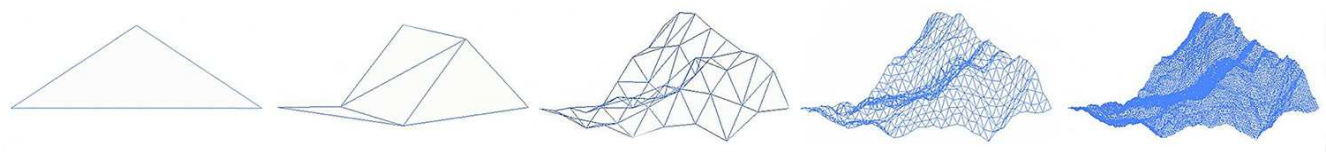
Eden od možnih pristopov h generiranju terena je uporaba funkcije dveh spremenljivk, katere graf je ploskev v trirazsežnem prostoru. Vendar se tak pristop navadno ne uporablja, saj so takšne funkcije težko izračunljive, niso dovolj naključne in njihovi grafi niso podobni pokrajinam v naravi.

V nadaljevanju bomo predstavili enega od možnih pristopov k naključnemu modeliranju terena z uporabo fraktalnega algoritma. Nastala fraktalna površina bo predstavljala želeni naključni teren. Zaradi fraktalnih lastnosti narave se izkaže uporaba takšnega algoritma kot zelo učinkovita, saj je takšno stopnjo naključnosti in kompleksnosti modela z uporabo klasičnih metod težko doseči. Na sliki 1 vidimo primer terena, generiranega s fraktalnim algoritmom.

Prelomni algoritem

S pojmom prelom opisujemo ploskovno razpoko ali razmeroma ozko razpokano cono v Zemljini skorji, povezano s preteklimi in morebitnimi prihodnjimi tektonskimi premiki [5]. *Fraktal* lahko definiramo kot geometrijski vzorec, ki ga dobimo s ponavljanjem nekega iterativnega procesa na preprostejšem mnogotniku ali poliedru.

Prelomni fraktal na grobo simulira posledice močnih potresov vzdolž naključnih prelomnih premic. Pod izrazom *prelomna premica* razumemo vsako takšno premico, ki gre skozi teren in ga deli na dva dela. Fraktal bomo ustvarili na prazni višinski sliki. *Višinska slika* je sivinska rasterska slika, ki se uporablja za shranjevanje vrednosti, običajno so to nadmorske višine. Bela barva predstavlja najvišjo višino,



SLIKA 1.
Primer generiranja fraktalnega terena v več korakih, od leve proti desni

črna pa najnižjo. Na prazni višinski sliki se bo izvršilo zaporedje »potresov« na sledeči način: na sliki se izbere naključna premica in vsako vozlišče nad to premico se premakne gor, vozlišča pod njo pa navzdol. Ta iterativni postopek se ponavlja, dokler ne dobimo zadovoljivega terena.

Psevdokoda

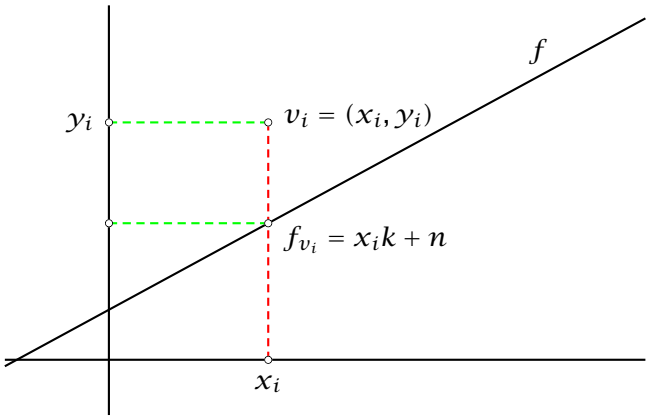
Algorithm 1 Prelomni fraktal

Require: Poligonska ravnina Π
Ensure: Poligonska mreža Π

```
1:  iter ⇐ 0
2:  while iter < stIteracij do
3:      k ⇐ naključno število
4:                          ▷ Zgenerira smerni koeficient k
5:      n ⇐ naključno število
6:                              ▷ Zgenerira odsek na osi y
7:      for all vozlišče v_i ∈ Π do
8:          f_{v_i} ⇐ x_i · k + n
9:                              ▷ Izračuna točko na premici
10:         if y_i > f_{v_i} then
11:                             ▷ Če je vozlišče nad premico
12:             dvigni z_i ⇐ z_i + Δh
13:                             ▷ Spremeni višino vozlišča v_i
14:         else
15:             spusti z_i ⇐ z_i − Δh
16:                             ▷ Spremeni višino vozlišča v_i
17:         end if
18:     end for
19:     zmanjšaj Δh ⇐ Δh− konst.
20:                             ▷ Zmanjša spremembo višine
21:     iter ⇐ iter + 1
22:                         ▷ Poveča števec izvedenih iteracij
23: end while
```

Delovanje algoritma

Poligonska ravnina je poseben primer poligonske mreže, ki zadošča Evklidovi definiciji ravnine. *Poligonska mreža* je v računalniški grafiki definirana kot 3-terica (V, E, F), kjer V predstavlja množico vozlišč (točk v prostoru), $E \subset (V \times V)$ množico robov (segmentov premice) in s $F \subset E^*$ označujemo množico lic (konveksnih poligonov). *Vozlišče* v računalniški grafiki predstavlja podatkovno strukturo, ki opisuje določene atribute. Običajno je to pozicija točke v



SLIKA 2.
Primer vozlišča v_i, ki leži nad generirano premico f.

dvorazsežnem ali trirazsežnem prostoru. Vozlišče v_i je 3-terica oblike (x_i, y_i, z_i), kjer so x_i, y_i in z_i elementi iz množice realnih števil $\mathbb{R}$.

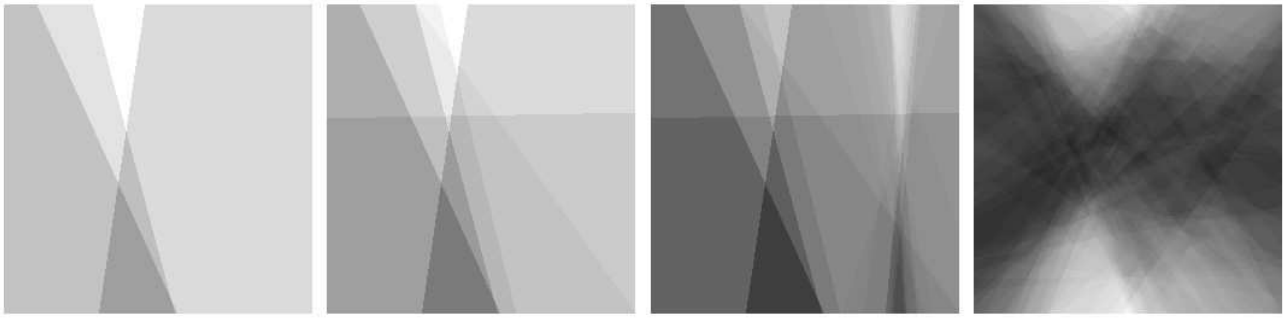
Poligonska ravnina se nahaja v tridimenzionalnem prostoru, vendar je premice dovolj generirati v dvodimenzionalnem prostoru. Zato je koordinata z v algoritmu uporabljena le takrat, ko se spremeni višina vozlišča v.

Algoritem izpolnjuje teren iterativno z izbiro dveh naključnih parametrov. Za vsako iteracijo se določita smerni koeficient k in odsek na y osi n.

$$y = k \cdot x + n \tag{1}$$

Enačba 1 predstavlja eksplicitno enačbo premice v ravnini $z = 0$, ki jo enolično določata parametra k in n. Za vsako vozlišče v_i poligonske ravnine Π je moč izračunati, ali v_i leži nad ali pod generirano premico. Algoritem preveri, ali bo vozlišče dvignil ali spustil tako, da koordinato x_i vstavi v formulo 1. Par (x_i, f_{v_i}) predstavlja točko, ki leži na premici f. Iz slike 2 je očitno, da zadošča pogledati odnos med koordinato y_i od vozlišča v_i in izračunan f_{v_i}. V primeru, da je $y_i \geq f_{v_i}$, se to vozlišče dvigne. Na tak način se preverijo vsa vozlišča, ki ležijo na poligonski ravnini, in priredi nova višina.

Po vsaki iteraciji se zmanjša sprememba višine Δh zato, da teren ne bo imel v vseh točkah enakih višinskih razlik. Atribut $consth$ je konstanta in določa razliko, za katero se zmanjšuje višina. Večja vrednost $consth$ pomeni hitrejše padanje Δh in s



SLIKA 3.
Primer štirih višinskih slik dimenzij 256×256 pikslov in barvne globine 8 po 4, 8, 32 in 128 iteracijah algoritma

tem tudi položnejši teren. Manjši $consth$ pomeni, da bo sprememba višine Δh manjša, zato se bo teren bolj strmo vzpenjal ali spuščal. Z drugimi besedami to pomeni, da $consth$ vpliva na razgibanost terena. Atribut $stIteracij$ neposredno vpliva na izgled terena, saj predstavlja število ponovitev algoritma. Za upodobitev realističnega terena je potrebnih veliko prelomov, kot je razvidno na sliki 5.

Sicer na takšen način ne moremo generirati navpičnih prelomov, saj v eksplicitni obliki naklon za navpično premico ni definiran, vendar izvzetje le–teh ne vpliva občutno na končni teren.

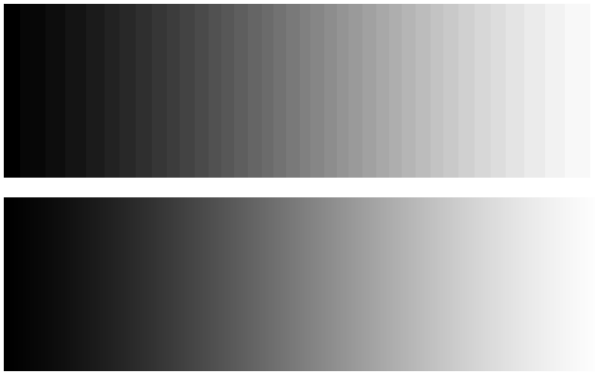
Primeri generiranih terenov

Na sliki 3 so prikazani primeri generiranih višinskih slik pri različnem številu iteracij prelomnega algoritma. Zgenerirana višinska slika je odvisna od števila zgeneriranih naključnih prelomov, dimenzije teksture ter barvne globine teksture.

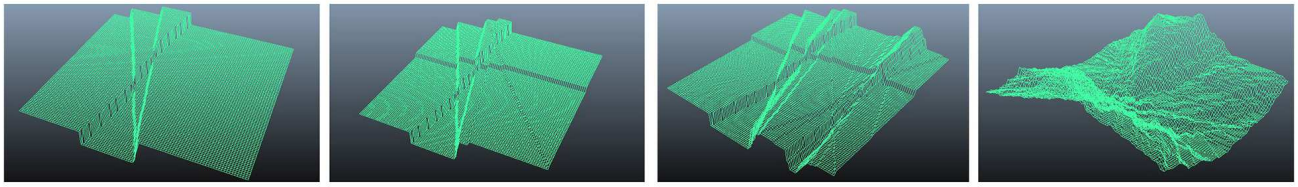
Na skrajno levi višinski sliki, prikazani na sliki 3, jasno ločimo, kje so potekale prelomne premice. Na sliki je le nekaj zelo podobnih sivih odtenkov, ki zavzemajo veliko površine. Iz tega lahko sklepamo, da bi teren iz takšne slike bil skoraj povsem raven, z malimi višinskimi razlikami. Druga slika je podobna prejšnji in pridemo tudi do enakih zaključkov. To ni presenetljivo, saj je razlika v iteracijah algoritma med slikama zelo majhna. Na tretjem primeru opazimo v zgornjem desnem kotu zabrisane meje, kjer so potekale prelomne premice. Ker je tisto območje svetlo obarvano, vemo, da bi v upodobitvi takšne višinske slike tam stal hrib, kot lahko vidimo na tretjem terenu na sliki 5.

Na skrajno desnem primeru zaznamo ogromno različnih odtenkov. Povemo lahko le, da ta višinska slika opisuje dve gori ter dolino med njima. Za razliko od prvih dveh višinskih slik, si je brez upodobitve v trirazsežnem prostoru takšen primer težko predstavljati.

Ker je višinska slika odvisna tudi od barvne globine generirane teksture, velja, da lahko teksture z večjo barvno globino tvorijo bolj razgibane terene.



SLIKA 4.
Primerjava sivinske lestvice med slikama z 8-bitno barvno globino (zgoraj) in 16-bitno barvno globino (spodaj)



SLIKA 5.
Primer generiranih terenov po 4, 8, 32 in 128 iteracijah algoritma na poligonski ravnini velikosti 10×10 enot z 10201 vozlišču v programu *Autodesk Maya*

Slika 4 ponazarja razliko dveh sivinskih lestvic.

Upodobitev višinskih slik iz slike 3 na ravnini v prostoru vidimo na sliki 5. Po dovolj veliko iteracijah algoritma lahko zmodeliramo realistični fraktalni teren, ki je dovolj dober za uporabo v računalniški grafiki ali na kakšnem drugem področju.

Zaključek

Vidimo, da fraktalni teren izkazuje značilnosti, ki bi jih pričakovali v pokrajinah iz narave. Sicer predstavljeni algoritem ne generira jam in votlin, za to se v praksi uporabijo posebni algoritmi, vendar pa smo z implementacijo prelomnega algoritma dobili realistične gore in doline. Če bi podoben teren želeli zmodelirati ročno, bi nam to vzelo precej časa.

Algoritem z vsako iteracijo obišče vsa vozlišča v poligonski mreži, zato je njegova časovna zahtevnost $O(n^2)$, kjer je n število vozlišč poligonske mreže. To med drugim pomeni, da algoritem ni primeren za uporabo v grafiki v realnem času, saj je prepočasen. Tovrstni algoritem se zato uporablja v predprocesiranju, kjer si vnaprej zgeneriramo poljubno število višinskih slik, ki jih potem uporabljamo kot teksture za upodobitev terena.

Postavi se nam vprašanje, ali je mogoče prelomni algoritem na kak način posplošiti. Izkaže se, da je z nekaj spremembami prelomni fraktal moč pretvoriti v algoritem, ki generira naključne planete. V tem primeru ne ločujemo ravnine s premico, ampak generiramo naključne ravnine, ki delijo sfero. Potrebno je samo določiti lego vozlišč glede na ravnino. Če se vozlišče nahaja nad prelomno ravnino, ga premaknemo v smeri njegovega normalnega vektorja, v primeru da se nahaja pod njo, pa v negativno smer tega vektorja.

Literatura

[1] D. S. Ebert, F. K. Musgrave, D. Peachey, K. Perlin in S. Worley, *Texturing and Modeling, Third Edition: A Procedural Approach*, Morgan Kaufmann, 2002.

[2] M. DeLoura, *Game Programming Gems 2*, Charles River Media, 2001.

[3] I. Parberry, *Tobler's First Law of Geography, Self Similarity, and Perlin Noise: A Large Scale Analysis of Gradient Distribution in Southern Utah with Application to Procedural Terrain Generation*, Technical Report LARC-2014-04, 2014.

[4] e-on software, inc. *Vue Helps Industrial Light & Magic Create Environments for »Pirates Of The Caribbean: Dead Man's Chest« VFX*, ogled 12. 1. 2016. Dostopno na naslovu: `http://www.e-onsoftware.com/news/?page=pressreleases\&date=August\%201,\%202006`.

[5] J. Lapajne. Uprava Republike Slovenije za zaščito in reševanje. *Nekateri tektonski, seizmotektonski in seizmološki termini – 1. Del*, ogled 12. 1. 2016. Dostopno na naslovu: `http://www.sos112.si/slo/tdocs/ujma/2008/316.pdf`.

×××