SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1599

Prikaz fraktalnog šuma višerezolucijskom kvadratnom rešetkom u stvarnom vremenu

Tomislav Tunković

Zagreb, lipanj 2018.

Umjesto ove stranice umetnite izvornik Vašeg rada.
Da bi ste uklonili ovu stranicu obrišite naredbu `\izvornik`.

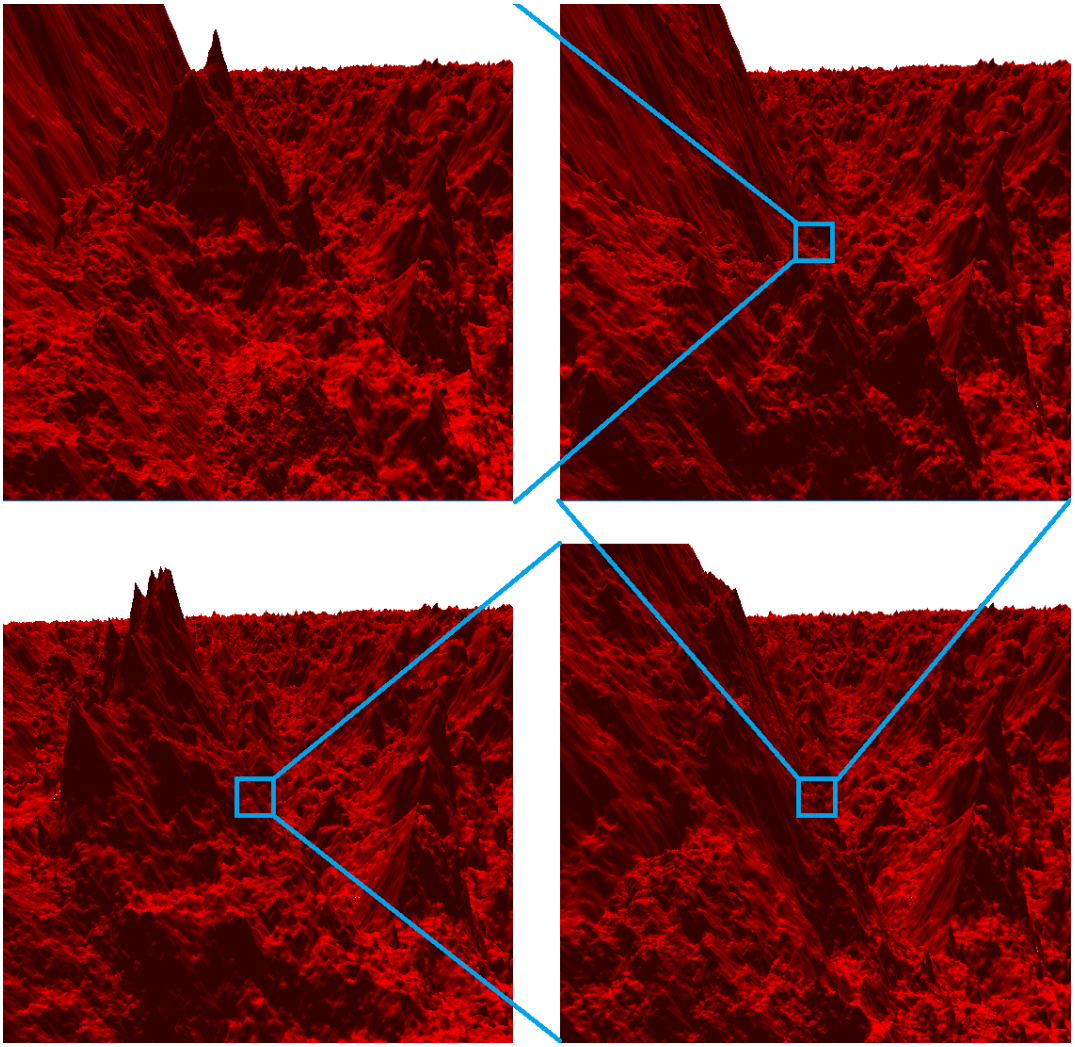
SADRŽAJ

POPIS SLIKA

1. Uvod

Fraktali su apstraktni objekti s beskonačnom hrapavošću i visokom razinom samosličnosti. Moguće im se približavati do proizvoljne udaljenosti, a da nikada ne ponestane detalja. U nekoj mjeri oni postoje i u prirodi, a uočavamo ih u biljkama, kristalima, morskim obalama i mnogim drugim fenomenima. Jedna od najčešćih primjena fraktala u računalnoj grafici je za generiranje terena, odnosno planina. Ovaj rad se bavi time, ali umjesto generiranja planina kakve vidimo u prirodi (slike 1.1), fokus je na apstraktnim planinama do kojih se može beskonačno približavati, i u njima otkrivati uvijek nove, manje planine, kao što je prikazano na slici 1.2.
Brzina tog približavanja je proporcionalna trenutnoj udaljenosti od fraktala. To znači da ako se promatrač kreće direktno prema površini, brzina će padati eksponencijalno kroz vrijeme. Svakih nekoliko sekundi će mu udaljenost do površine biti duplo manja, ali ju nikada neće dotaknuti. Što se više približi, to će više sitnijih planina moći vidjeti, a prethodne će ostajati u pozadini osim ako ih ne prekriju nove.

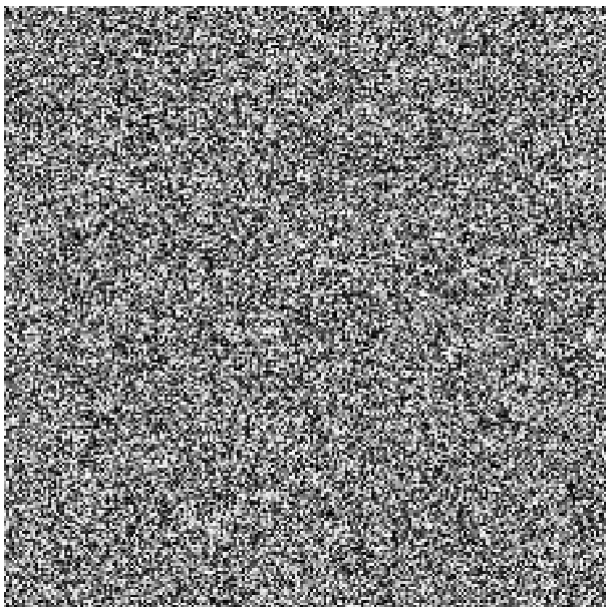
Slika 1.1: Prirodne planine



Slika 1.2: Nekoliko razina uvećanja

2. Fraktalni šum

Definirajmo diskretni uniformni 2D šum kao $u : \mathbb{Z}^2 \to [-1, 1\rangle$. Vizualizacija jednog kvadratnog dijela domene te funkcije se može napraviti slikom u kojoj su pikseli nijanse sive boje. -1 odgovara crnoj, a 1 bijeloj. To je prikazano na slici 2.1.



Slika 2.1: Uniformni šum

Jedna jednostavna varijanta diskretnog fraktalnog šuma se može definirati kao zbroj više uniformnih gdje je amplituda svakog idućeg dvostruko manja, a frekvencija dvostruko veća [2].

$$f : \mathbb{Z}^2 \to [-1, 1\rangle$$

$$f(p) = \frac{1}{2} \sum_{i=0}^{n} \frac{1}{2^i} u(\frac{p}{2^{n-i}})$$

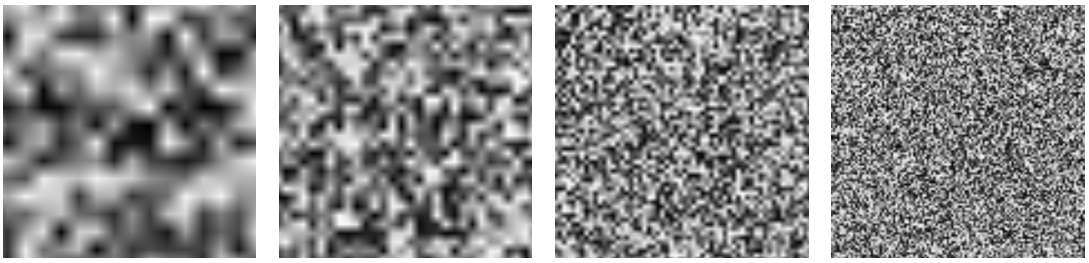
Ovdje se u koristi s realnim argumentom, pa se u tom slučaju definira kao bilinearna

interpolacija najbliže četiri vrijednosti:

$$u(p) = mix(mix(u(\lfloor p \rfloor + (0,0)), u(\lfloor p \rfloor + (1,0)), frac(p_x)),$$
$$mix(u(\lfloor p \rfloor + (0,1)), u(\lfloor p \rfloor + (1,1)), frac(p_x)), frac(p_y))$$

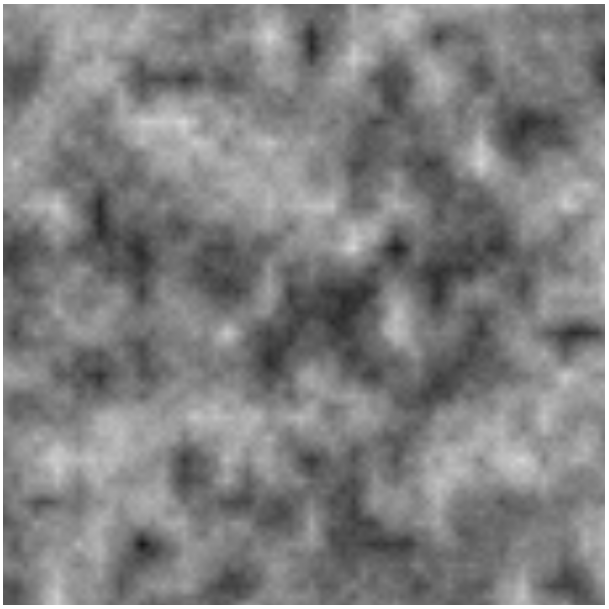
$$frac(x) = x - \lfloor x \rfloor$$

$$mix(a, b, x) = a \cdot (1 - x) + b \cdot x$$



(a) Oktava 0 **(b)** Oktava 1 **(c)** Oktava 2 **(d)** Oktava 3

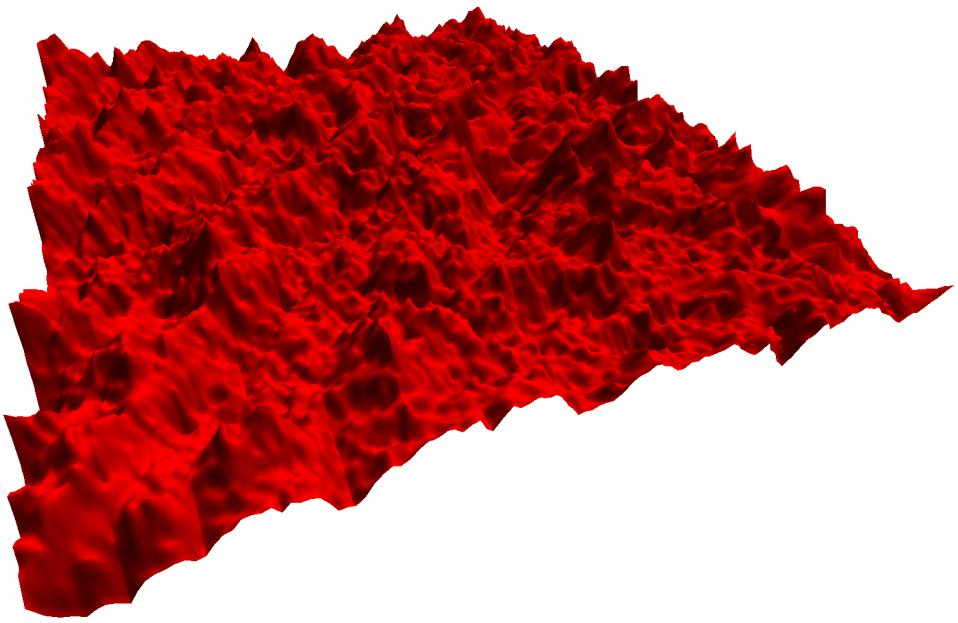
Slika 2.2: Pojedine oktave bez skaliranja amplitudom



Slika 2.3: Suma svih oktava

Slika 2.2 prikazuje četiri oktave uniformnog šuma koje kad se skaliraju s odgovarajućim amplitudama i zbroje daju rezultat prikazan na slici 2.3. Drugi način kojime se može vizualizirati takav 2D šum je tako da se iskoristi kao visinska mapa za kvadratnu mrežu. Svaki vrh mreže će se po vertikali pomaknuti za iznos proporcionalan onome u visinskoj mapi. Takav prikaz je manje praktičan i računski bitno zahtjevniji, ali je

zato vizualno puno zanimljiviji zbog svoje trodimenzionalnosti. Primjer toga se može vidjeti na slici 2.4. Trokute je naravno nužno sjenčati na neki način kako ne bi sve bilo potpuno identične boje.



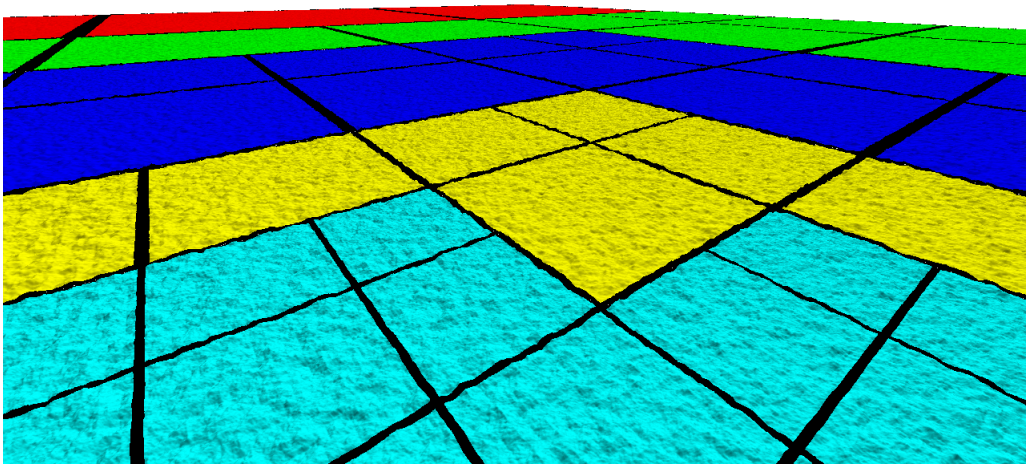
Slika 2.4: Kvadratna mreža s visinskom mapom

3. Prikaz višerezolucijskom kvadratnom mrežom

Kako bi se zadržala razina detalja na slici prilikom približavanja površini kvadratne mreže potrebno je povećavati broj vrhova mreže. U suprotnom bi se naravno moglo približiti jednom od kvadrata i njegova ravna površina bi bila jedino što bi se vidjelo. Svakom idućom razinom detalja se pribraja još jedna oktava fraktalnog šuma i udvostručava broj vrhova po dimenziji kvadratne rešetke. Time eksponencijalno rastu zahtjevi za računalnim resursima koje vrlo brzo neće biti moguće zadovoljiti.

Međutim, nisu svi dijelovi rešetke jednako blizu, a za one koji su udaljeniji nam treba i manje vrhova da bi postigli jednaku razinu detalja na ekranu. Idealno bi bilo da je gustoća vrhova rešetke na ekranu otprilike konstantna bez obzira koliko udaljeni dio rešetke se promatra. Radi jednostavnosti se zanemaruje kut s normalom i potencijalna preklapanja različitih dijelova rešetke, već promatramo kao da je sve okomito na smjer gledanja.

Da bi se moglo iskoristiti tu obzervaciju prvo je potrebno rešetku particionirati na pločice. Svaka pločica će imati jednak broj kvadrata, recimo 128×128, ali mogu biti različitih dimenzija. Te dimenzije će uvijek biti oblika $a \cdot 2^l$ gdje je a neka konstanta, a l razina detalja. Particiju kompletne rešetke se radi tako da su veličine pločica otprilike proporcionalne udaljenosti od promatrača [1]. To će izgledati onda kao što je prikazano na slici 3.1.

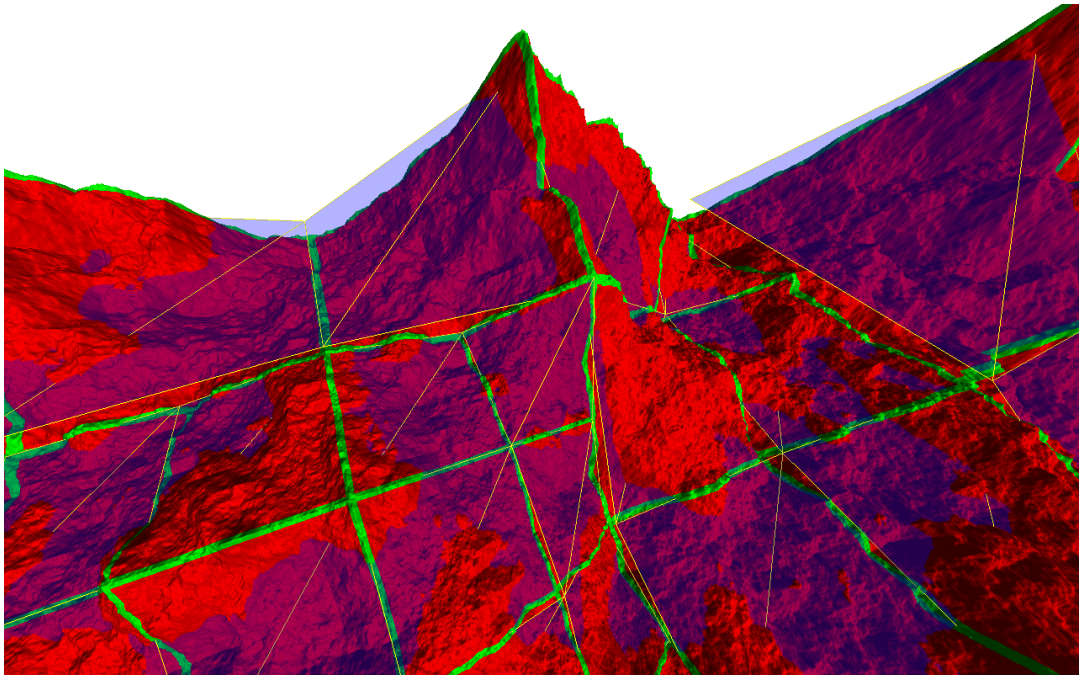


Slika 3.1: Pločice različitih veličina

Pločice su organizirane u hijerarhiju, odnosno u kvadratno stablo. Korijen stabla odgovara pločici koja pokriva cijelu površinu koja se vizualizira, a njegovih četvero djece svaki po svoju četvrtinu i tako dalje za svaki idući čvor stabla. Poziva se rekurzivna funkcija koja se spušta kroz stablo kako bi se odlučilo koje pločice se crtaju. Da bi se jedna pločica crtala mora biti zadovoljen slijedeći uvjet:

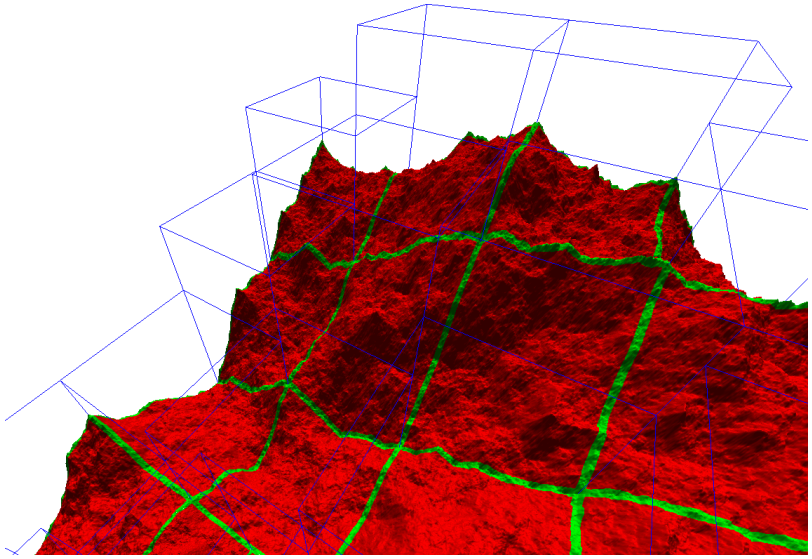
$$\frac{udaljenost\_promatraca\_do\_plocice}{udaljenost\_izmedu\_susjednih\_vrhova\_mreze} > c$$

Ako se odluči da se pločica crta, onda se rekurzija više ne zove dalje. Za izračun udaljenosti do pločice nužno je imati u CPU RAM-u nekakve podatke o visini vrhova njenog dijela mreže. Ti podaci bi se mogli preuzimati iz GPU RAM-a, ali onda postoji opasnost da se jako uspori rad grafičke kartice. Zato se visinska mapa računa i na procesoru za svaku pločicu. To se ne radi u punoj rezoluciji koja bi mogla biti veličine 128×128, nego samo 2×2. Dakle visina će se izračunati samo u kutovima pločice na najmanjoj mogućoj razini detalja. Slika 3.2 prikazuje plave prozirne kvadrate čije su visinske mape izračunate na CPU na puno nižoj rezoluciji.



Slika 3.2: CPU visine

To je dovoljno informacija za dobiti omeđujuće kutije pločica. Izračuna se minimum i maksimum od te četiri vrijednosti i taj se raspon dodatno proširi za sumu svih amplituda šuma koje još nisu pribrojane. Suma svih već nepribrojenih amplituda je konačna vrijednost jer se radi o geometrijskom redu. Između tih granica će se nalaziti cijela visinska mapa unutar te pločice. Za procjenu minimalne udaljenosti do pločice se uzima udaljenost do te kutije. One ne moraju savršeno dodirivati pločice po vertikalnoj komponenti, kao što nije niti slučaj na slici 3.3, ali je bitno da sadržavaju sve vrhove koji se nalaze unutra, uključujući i one koji će se vidjeti tek nakon dodavanja viših oktava šuma.



Slika 3.3: Omeđujuće kutije pojedinih pločica

Nakon što su odabrane sve pločice koje moraju biti nacrtane, potrebno je izračunati visinsku mapu na njihovom području i rezoluciji. To se radi pomoću visinske mape roditeljske pločice tako da se uzme dio koji pokriva trenutnu pločicu i bilinearno naduzorkuje te nadoda iduća oktava šuma s dvostruko manjom amplitudom. Rezultat se sprema u jedan od slojeva niza tekstura u memoriji grafičke kartice. Jednom kada je to izračunato, to se zapamti i čuva dok god je pripadajuća pločica potrebna ili bilo koja od njezinih potomaka.

4. Beskonačno uvećanje

Prethodno opisan pristup će funkcionirati do određene razine uvećanja, ali će se brzo početi pojavljivati različiti problemi. Jedan dio njih se svodi na problem konačne preciznosti zapisa decimalnih brojeva u računalu. Zbog toga je nužno paziti kako se koriste decimalni brojevi i voditi računa o tome koliko je decimala potrebno za izračune. S obzirom da se radi binarnom zapisu s mantisom i eksponentom, niti broj bitova u eksponentu u nekom trenutku više neće biti dovoljan.
Drugi problem su memorijski zahtjevi. Što se više promatrač približava terenu, potreban je sve veći broj pločica za prikaz detalja oko promatrača, a memorija je naravno ograničena. Osim toga što nam treba više memorije za pohranu pločica, potrebno je i vrijeme da se sve one nacrtaju na ekran. Ako njihov broj samo raste onda će u nekom trenutku postati presporo.

4.1. Neslaganja CPU i GPU izračuna

Podaci o visinskim mapama se računaju odvojeno na GPU i CPU. Na GPU za prikaz na ekranu, a na CPU za računanje udaljenosti. Ako se savršeno ne poklapaju vrijednosti izračunate i na GPU i na CPU, onda će nakon dovoljno velikog uvećanja te dvije visinske mape u potpunosti divergirati. To se događa jer se za iduću razinu detalja koriste podaci iz prethodne. Ako su podaci iz prethodnih različiti, onda se greška još više povećava u idućima. Procjena udaljenosti na CPU tada neće odgovarati onome što se vidi na ekranu pa će biti selektirane pogrešne rezolucije pločica za crtanje.

Jedan uzrok tom problemu je interpolator na grafičkoj kartici koji računa vrijednosti ulaza u piksel sjenčar, pritom interpolirajući izlaze iz sjenčara vrhova. Rezultati koji se tako dobivaju su često neprecizni, čak i u slučajevima kada postoji dovoljno bitova da se rezultat zapiše savršeno. Zbog toga je potrebno napraviti korekciju u sjenčaru piksela zaokruživanjem na najbližu očekivanu vrijednost. U slučaju računanja visinskih mapa, potrebna je teksel koordinata u roditeljskoj visinskoj mapi pa korek-

cija izgleda ovako:

$$parentPix \leftarrow round(parentPix \cdot 2) \cdot \frac{1}{2}$$

Time se osigurava da će se $parentPix$ uvijek nalaziti točno na sredini ili na rubu piksela u roditeljskoj visinskoj mapi. Još jedan izvor problema je automatski bilinearni interpolator za uzorkovanje tekstura. On u nekoj fazi izračuna koristi premali broj bitova pa rezultat ispadne zaokružen. Zbog toga je nužno ručno uzorkovati četiri susjedna piksela i napisati vlastiti kod koji će to bilinearno interpolirati.

Posljednji problem koji nastaje je prilikom aritmetike s decimalnim brojevima. Ako se računa nešto i cijeli rezultat ne stane u zapis s pomičnim zarezom, nužno je napraviti zaokruživanje. To zaokruživanje može biti napravljeno drugačije na CPU i GPU. Negdje će se 1 u binarnom zapisu koji ne stane u bitove mantise zaokružiti prema gore, a drugdje prema dolje. Taj problem se najlakše može riješiti tako da se ručno zaokružuje na predvidivi način funkcijom $floor$ prije nego što dođe do nepredvidivog automatskog zaokruživanja. Konkretno u ovom slučaju se u sjenčaru piksela za izračun visinskih mapa konačna vrijednost zaokružuje na slijedeći način:

$$h \leftarrow \lfloor h \cdot floorMul \rfloor \cdot floorMul^{-1}$$

$$floorMul = 2^k$$

Eksponent k za $floorMul$ mora biti određen tako da ne dolazi do automatskog zaokruživanja kada se rezultat koristi kasnije za izračun visinske mape djeteta. Također treba paziti da se ne koriste nikakve kompleksnije operacije koje mogu producirati broj koji će biti automatski zaokružen. Zato su svi izračuni vrlo jednostavna zbrajanja i množenja. Bilinearnom interpolacijom ako su interpolacijski faktori uvijek ili 0 ili 0.5, moguće je dobiti najviše dva dodatna bita rezultata u odnosu na ulaze. To se događa u slučaju kada su oba faktora 0.5 pa se svaki od ulaza množi s ukupno 0.25. To je glavni razlog zašto je ručno zaokruživanje potrebno na kraju sjenčara.

4.2. Relativizacija

Kako bi se omogućilo beskonačno uvećanje, nužno je podržati pločice proizvoljno malih veličina. Ako je neka pločica u apsolutnom prostoru udaljena od ishodišta za 1, a veličina joj je 2^{-30}, onda se mora moći prikazati broj $1 + 2^{-30}$ u zapisu s pomičnim zarezom. To očito nije moguće jer broj bitova u mantisi nije dovoljan. Međutim, apsolutna pozicija pločice nije potrebna kako bi ju se moglo nacrtati, nego samo pomak

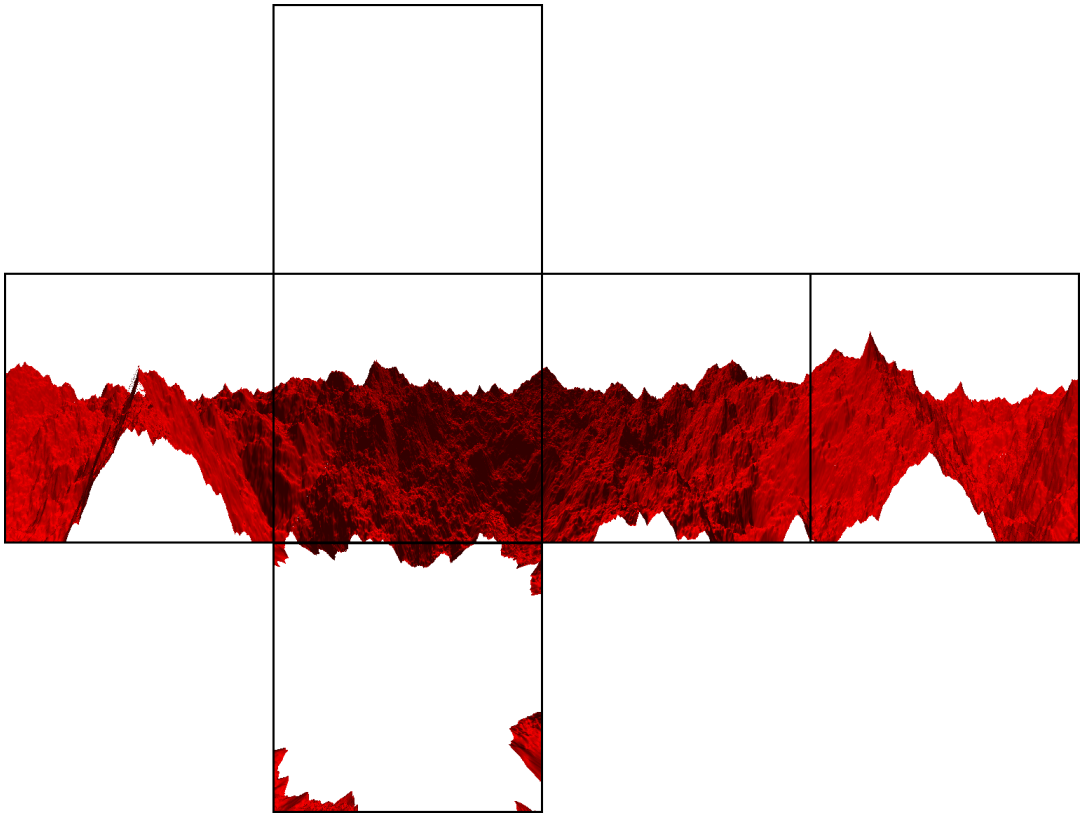
u odnosu na promatrača. Red veličine tog pomaka ne može biti neograničeno mnogo različit od veličine pločice koju se crta. To je kriterij po kojemu ih se selektira. Dakle, preciznost zapisa s pomičnim zarezom je dovoljna ako se prilikom rekurzivnog spusta kroz hijerarhiju pločica pamti koliki je apsolutni pomak u odnosu na trenutnu pločicu, i svakim novim pozivom za pločicu dijete pridoda razlika njenog pomaka u odnosu na roditelja i pomaka pločice na istoj razini u kojoj je promatrač. Horizontalne x i z komponente pomaka su jednostavne za računati, ali vertikalna y je složenija jer ovisi o visinskoj mapi pločice. Za y pomak pločice se uzima srednja vrijednost omeđujuće kutije zaokruženo na višekratnik veličine pločice u horizontalnom smjeru. Visinske mape također moraju biti relativne u odnosu na svoje roditelje zato da bi im se vrijednosti kretale oko 0. Osim toga, raspon im treba ostati oko intervala od -1 do 1, a ne težiti u 0 jer bi onda ponestalo bitova u eksponentu za zapisati ih. Zbog toga se roditeljske visine množe s 2, a amplituda šuma koji se pribraja je uvijek 1 u lokalnom prostoru pojedine pločice. Kada se pločica crta, vrijednosti visina je naravno nužno pomaknuti za iznos koji je dobiven kao razlika pomaka kamere i cijele pločice, ali i pomnožena s odgovarajucim 2^k kako bi se visine skalirale iz lokalnog u apsolutni prostor.

4.3. Dubinski spremnik

Klasičan Z dubinski spremnik će vrlo brzo ostati bez preciznosti za prikaz raspona udaljenosti koje su ovdje potrebne. Kako bi se poboljšala preciznost, koristi se obrnuti 32bit decimalni dubinski spremnik [3]. Obrnuti znači da se bliska i daleka ravnina zamjene. Umjesto klasičnog 24bit cijelobrojnog formata koristi se 32bit decimalni jer se onda logaritamska distribucija decimalnih brojeva u zapisu s pomičnim zarezom poništi sa z dijeljenjem i dobije se rezolucija koja linearno pada s dubinom. To je točno ono što je potrebno u slučaju ovog rada. Problem s OpenGL-om je taj što su standardno ravnine za dubinsko rezanje na $z = -1$ i $z = 1$. To znači da ako se koristi dubinski spremnik s pomičnim zarezom onda je 0 na sredini intervala, a oko 0 je najveća preciznost. Time je izgubljena sva prednost dobivena tim formatom. Na sreću, od OpenGL-a 4.5 postoji funkcija $glClipControl$ kojom se raspon može namjestiti od 0 do 1.

4.4. Pozadina

Nakon nekog uvećanja, čak i obrnuti dubinski spremnik s 32bit zapisom s pomičnim zarezom neće biti dovoljan. Osim toga, broj pločica koje treba nacrtati će isto narasti pa ih se puno mora čuvati u memoriji i trošiti performanse na njihovo crtanje. Međutim, tada je uvećanje već toliko veliko da se one najveće pločice u pozadini više uopće ne pomiču jer je brzina kretanja promatrača premala u odnosu na to koliko su one udaljene. Što je manja udaljenost promatrača do površine i što se on sporije kreće, sve je više pločica u daljini moguće nacrtati u kockastu pozadinsku mapu. Kada se otvore stranice te kocke onda ona izgleda kao na slici 4.1. Pozadinsku mapu tada koristimo za crtanje u jednom prolazu svih tih pozadinskih pločica na ekran. Time je riješen problem performansi s rastućim brojem pločica koje se crtaju. Problem s rasponom dubinskog spremnika se rješava promjenom korijenskih pločica u odnosu na koje se sve računa. Nakon što su sve razine pozadinskih pločica nacrtane u pozadinsku mapu, prva iduća razina postaje korijenska. Time je zamijenjen trenutni apsolutni prostor za skalirani koji puno bolje odgovara rasponu udaljenosti koje su potrebne za trenutne pločice. Ako se promatrač više neće vraćati nazad, na manje razine uvećanja, onda pločice koje su nacrtane u pozadinsku mapu više nisu potrebne i mogu se odbaciti. To znači da je problem rasta memorijskih zahtjeva s uvećanjem riješen. S tom pretpostavkom je nužno čuvati samo pločice sa zadnjih nekoliko razina uvećanja, kojih nikada neće biti više od neke konstante.



Slika 4.1: Pozadinska kockasta mapa

4.5. Povratak iz uvećanja

Crtanjem pločica u pozadinsku mapu i brisanjem visinskih mapa iz memorije se gubi mogućnost povratka na manje razine uvećanja jer ih nije moguće rekonstruirati iz djece. Moguće bi bilo raditi rekonstrukciju iz korijenske pločice iz koje je sve poteklo, ali ako se promatrač već jako približio površini, morat će se rekonstruirat jedan jako dugi niz pločica što može potrajati duže vrijeme. Kako bi se ipak zadržala mogućnost povratka nužno je zapamtiti te pločice na neki način. Jedna opcija je zapamtiti samo četiri visine u kutovima pločica na odgovarajućoj rezoluciji, ona ista koju već imamo u CPU RAM-u. Iz tih podataka je moguće rekonstruirati originalnu rezoluciju bilinearnim naduzorkovanjem i zbrajanjem preostalih oktava šuma. Drugi problem predstavlja pozadinska mapa. Nakon što se promatrač dovoljno pomakne od originalnog mjesta gdje je nacrtana pozadinska mapa, ona više neće predstavljati ono što bi se vidjelo kada bi se pločice direktno crtale na ekran. To bi značilo da svakih nekoliko razina uvećanja se treba zapamtiti kako je izgledala pozadinska mapa. S obzirom da su to kockaste mape i da moraju biti dovoljne rezolucije da se crtaju preko cijelog ekrana, bez da se vidi pikselizacija, one će zauzimati jako puno prostora u memoriji. Zbog toga ih je

nužno spremati na što je moguće većim razmacima u razinama uvećanja. Međutim, ti razmaci ne mogu biti preveliki zbog ograničenja preciznosti dubinskog spremnika i vremena potrebnog da se jedna po jedna na ekran crtaju velike količine pločica. Zbog toga je za jako duboka uvećanja nužno te pozadinske mape preuzimati iz GPU RAM i prebacivati na CPU RAM, a kasnije i na HDD ili SSD. To sve bi se trebalo događati unaprijed i asinkrono kako simulacija ne bi zastajkivala kada se tako velike teksture prebacuju između različitih tipova memorije.

5. Performanse

Program je testiran na računalu s 970 GTX grafičkom karticom, na rezoluciji 2560×1440. Radi bez problema na 60 FPS. Vrijeme potrošeno na CPU za selekciju pločica koje treba nacrtati i poziv svih GPU API naredbi je zanemariv. Vrijeme potrošeno na GPU za crtanje pločica na ekran je do $5ms$, a računanje visinskih mapa je zanemarivo. Jedini problem nastaje kada uvećanje dovoljno naraste pa se puno pločica mora nacrtati u pozadinsku mapu odjednom. U tom trenutku se osjeti blagi trzaj, ali to se ne događa jako često i nije pretjerano uočljiv. To bi moglo biti rješivo tako da se crtanje pločica u pozadinski spremnik produlji kroz nekoliko okvira.
Daleko najviše memorije troše pozadinske RGB kockaste mape. Točan iznos ovisi o nekoliko različitih postavki. S najvišim postavkama su im dimenzije 2048 i kreira se nova za svaki faktor uvećanja od 2^{10}.

6. Zaključak

Rad demonstrira da je moguće u stvarnom vremenu ostvariti beskonačno uvećanje u 2D kvadratnu mrežu s visinskom mapom generiranom pomoću fraktalnog šuma, ako se onemogući povratak na prijašnje razine uvećanja. Za povratak na prijašnje razine je neizbježna $O(n)$ memorijska složenost, gdje je n razina uvećanja.
Vizualni efekt koji se postiže eksponencijalnim približavanjem i udaljavanjem je vrlo zanimljiv, pomalo hipnotičan, a ponekad i dezorijentirajući. Moguće je spustiti se od skale na kojoj se vidi nešto veličine planeta Zemlje pa sve do elektrona, i onda ponovo do elektrona kad bi zamislili da je onaj prvi veličina Zemlje, pa ponovo unazad do originalne Zemlje. To je nešto na što ljudski mozak nije navikao. Jedan eksperiment koji bi vrijedilo probati je napraviti podršku za VR HMD pa doživjeti to na puno izravniji način.

LITERATURA

[1] F. Strugar, “Continuous distance-dependent level of detail for rendering heightmaps (cdlod).” [Online]. Available: http://www.vertexasylum.com/downloads/cdlod/cdlod_latest.pdf

[2] H. Elias, “Perlin noise.” [Online]. Available: https://web.archive.org/web/20080724063449/http://freespace.virgin.net/hugo.elias/models/m_perlin.htm

[3] N. Reed, “Depth precision visualized,” 2015. [Online]. Available: https://developer.nvidia.com/content/depth-precision-visualized

[4] P. Decaudin, “Anttweakbar.” [Online]. Available: http://anttweakbar.sourceforge.net/doc/

Dodatak A
Upute za korištenje programa

A.1. Kontrole

- F - aktivacija i deaktivacija kontrola kamere
- W - naprijed
- S - nazad
- A - lijevo
- D - desno
- Space - gore
- C - dolje
- Q - kotrljanje lijevo
- E - kotrljanje desno
- Miš - pomicanje pogleda
- Kotačić - kontrola brzine kretanja
- F1 - ukljuvčivanje i isključivanje korisničkog sučelja

A.2. Korisničko sučelje

Korisničko sučelje je napisano pomoću AntTweakBar-a [4]. U lijevom panelu se mogu uključivati dodatne opcije za vizualizaciju. Na gornjem desnom panelu se nalaze vremenska mjerenja pojedinih aspekata programa. Svaki od njih se može otvoriti i tamo se vidi njegova boja. Ta mjerenja su prikazana kroz vrijeme na grafu ispod. Dodatna objašnjenja se mogu pronaći pritiskom na gumb '?' skroz dolje lijevo.

Prikaz fraktalnog šuma višerezolucijskom kvadratnom rešetkom u stvarnom vremenu

Sažetak

Predstavljena je jednostavna varijanta 2D fraktalnog šuma. Demonstrirano je kako se takav šum može iskoristiti kao visinska mapa za 2D kvadratnu rešetku. Objašnjen je algoritam kojime se kvadratna rešetka particionira na pločice različitih veličina ovisno o udaljenosti od promatrača. Riješeni su svi problemi koji nastaju naivnom implementacijom u slučaju kada se pokušava jako približavati površini i omogućeno beskonačno uvećanje, ali bez mogućnosti povratka. Ponuđene se neke metode kako uz linearnu memorijsku složenost zadržati mogućnost povratka.

Ključne riječi: Fraktali, uvećanje, kvadratna rešetka, stvarno vrijeme

Real time fractal noise rendering using multiresolution square grid

Abstract

Simple variation of 2D fractal noise is presented. It is demonstrated how such noise can be used as a height map for 2D square grid. Algorithm is explained for partitioning the entire square grid into tiles of different sizes based on how far from the viewer they are. All problems of a naive implementation related to getting very close to the surface are solved, enabling infinite zoom in, albeit without the possibility of zooming out again. Some linear memory complexity methods are considered for preserving the ability to zoom out.

Keywords: Fractals, zoom, square grid, real time