SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 446

Postupci prikaza terena

Karlo Jež

Zagreb, lipanj 2012.

SADRŽAJ

1. Uvod

Jedan od najčešćih motiva za prikazivanje u računalnoj grafici su otvoreni prostori, a najvažniji i najvidljiviji dio otvorenog prostora je površina terena.

U ovom radu dan je kratak pregled nekih tehnika generiranja, spremanja i prikazivanja terena. Postoji čitav niz metoda za prikazivanje stvarnih i zamišljenih terena, a ovaj rad će pokriti neke od najčešće korištenih. Prikaz terena često se koristi zajedno s prikazom drugih objekata poput drveća, zgrada, vodenih površina i slično pa je napravljen kratak pregled prikaza dodatnih vrsta objekata poput raslinja i vode.

Odabrane su dvije osnovne vrste terena: tereni zasnovani na visinskim mapama i tereni zasnovani na volumnim mapama. Sve implementirane tehnike napravljene su da budu izvedive u stvarnom vremenu i da uzimaju u obzir memorijska ograničenja današnjih računala i grafičkog sklopovlja, a neke su radi još većeg ubrzanja implementirane na samom grafičkom sklopovlju.

U prvom poglavlju dan je opis terena zasnovanih na visinskim mapama, s naglaskom na metode generiranja i teksturiranja terena. Slijedi poglavlje o terenima zasnovanim na volumnim mapama. Zadnje poglavlje govori o implementaciji obaju vrsta terena, implementaciji detalja na terenu poput visoke trave, kao i o prikazivanju reflektivne vodene površine. Kraj poglavlja i rada posvećen je prikazu dodatnih efekata koji mogu poboljšati vjernost prikaza terena.

2. Tereni zasnovani na visinskim mapama

2.1. Visinska mapa

Visinska mapa je dvodimenzionalno polje koje sadrži niz vrijednosti. Indeksi tog polja označavaju položaj točke na ravnini xz, a vrijednosti pohranjene u polju označavaju položaj na osi y. Veličine predstavljene vrijednostima visinske mape su proizvoljne, mogu predstavljati mikroskopske dimenzije, ali i veličinu kontinenata.

Visinske mape predstavljaju vrlo učinkovit način spremanja podataka o terenu. Potrebno je pohranjivati samo vrijednosti visine, jer je položaj na ravnini xz zadan implicitno. Visinska se mapa stoga sprema kao jednobojna dvodimenzionalna slika pri čemu svjetlina svake točke odgovara njezinoj visini. Jedan kanal boje u standardnom 32-bitnom RGBA zapisu ima osam bitova i može spremiti 256 (2^8) diskretnih vrijednosti. To nije dovoljno za terene koji imaju veće izbočine, poput planina, ali i sitnije detalje. Rezultat je diskretiziran i stepeničast izgled terena. Korištenjem sva četiri kanala (32 bita) može se spremiti 2^{32} diskretnih vrijednosti i na taj način prikazati i veće i sitnije detalje.

2.2. Generiranje visinske mape

Za generiranje visinske mape mogu se koristiti razni algoritmi za generiranje šuma ili fraktalni algoritmi. Također, može se kombinirati veći broj oktava šuma te tako dobiti fraktalni šum. Osim pseudoslučajnih metoda za generiranje terena moguće je koristiti ručno napravljene slike koje predstavljaju visine, kao i slike koje slučajnom algoritmu kod generiranja terena služe kao „smjernica“ (npr. korisnik na slici određuje opće karakteristike terena poput vode, ravnica ili planina, a algoritam generira teren nalik onom nacrtanom).

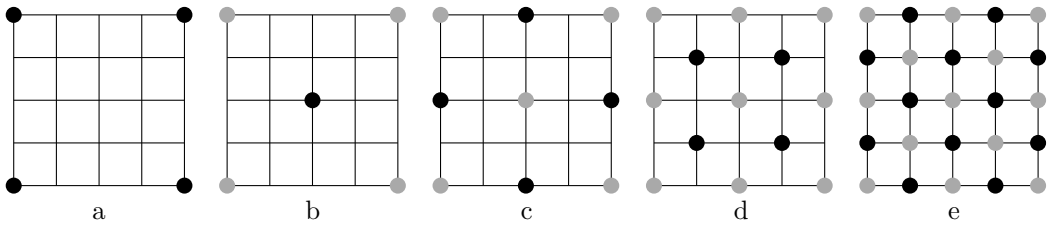
2.2.1. Algoritam dijamant-kvadrat

Algoritam dijamant-kvadrat je fraktalni algoritam temeljen na pomaku srednje točke. Stranice mape moraju biti veličine $2^n + 1$, primjerice 129 x 129, jer u svakom koraku algoritma određujemo vrijednost srednje točke koja u svakoj iteraciji sigurno postoji samo kod takve dimenzije terena. Za inicijalizaciju algoritma četiri rubne točke (uglovi) postavljaju se na definirane ili slučajne vrijednosti. Nakon toga se u svakoj iteraciji algoritma vrše dva koraka, dijamant i kvadrat.

Dijamant: Središnja se točka kvadrata postavlja na srednju vrijednost četiri rubne točke te joj se dodaje slučajna količina šuma (pozitivna ili negativna).

Kvadrat: Određuju se četiri srednje točke na stranicama između uglova, postavlja ih se na njihovu srednju vrijednost i dodaje im se malo manja količina šuma nego u prethodnom koraku.

Na ovaj način definirali smo rubne točke četiri kvadrata dimenzija stranice $2^{n-1}+1$ kojima su poznate jedino vrijednosti uglova. Isti postupak se rekurzivno ponavlja za sva četiri dobivena kvadrata sve dok $n > 0$.



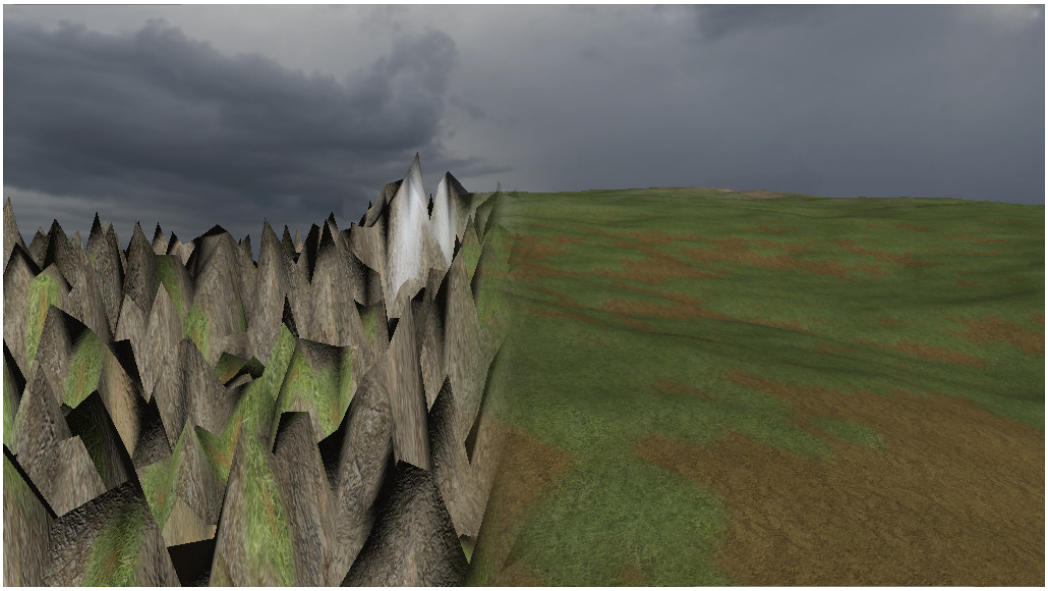
Slika 2.1: Prikaz rada algoritma dijamant-kvadrat na mapi veličine 5 x 5.

Na slici 2.1 možemo vidjeti prikaz rada algoritma. Sličica a prestavlja početno stanje algoritma s četiri definirana ugla. Na sličici b prikazan je korak dijamant. Uglovi s točkom u sredini definiraju kvadrate zarotirane za $45°$, tj. dijamante. Sličica c prikazuje korak kvadrat. Na sličici d vidimo korak dijamant proveden na četiri manja kvadrata dobivena u prošlom koraku. Na posljednoj sličici vidi se kraj rada algoritma.

Za svaki stupanj dubine rekurzije potrebno je smanjiti količinu slučajnog šuma da bi se dobio vjerniji prikaz terena jer očekujemo manje razlike u visini kod susjednih točaka terena (slika 2.2).

2.2.2. Perlinov šum

Perlinov šum je funkcija za generiranje koherentnog šuma u prostoru. Koherentan šum znači da se šum između bilo koje dvije točke u prostoru mijenja glatko, tj. bez diskontinuiteta.



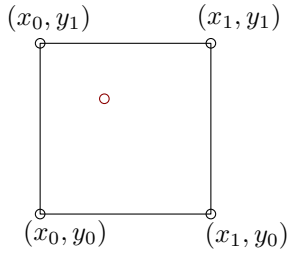
Slika 2.2: Prikaz rada algoritma sa i bez smanjivanja amplitude šuma u svakom koraku rekurzije.

Perlinov šum je preslikavanje bilo koje točke u prostoru u realan broj u intervalu $[-1, 1]$. Generirani šum je pseudoslučajan, za isti parametar uvijek daje istu vrijednost na izlazu.

$$f : \mathbb{R}^n \to x, \quad x \in \mathbb{R},\ x \in [-1, 1]$$

Ovdje ćemo opisati način rada dvodimenzionalnog Perlinovog šuma. Ponašanje je slično za jednu, ali i za više dimenzija.

Funkcija za generiranje šuma definirana je na mreži, pri čemu su točke mreže cijeli brojevi, dok se necijeli decimalni brojevi nalaze između točaka mreže. Pretpostavimo da tražimo vrijednost 2D funkcije šuma $noise2d(x, y)$, a ni x ni y nisu cijeli brojevi, tj. točka (x, y) se nalazi unutar kvadrata mreže. Prvo se određuju vrijednosti točaka mreže koje okružuju točku (x, y). Na 2D ravnini, imamo četiri takve točke (U 3D prostoru bilo bi ih osam). Nazovimo ih (x_0, y_0), (x_0, y_1), (x_1, y_0) i (x_1, y_1).

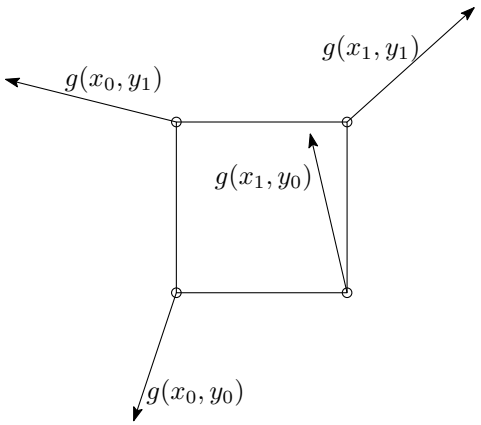


Slika 2.3: Četiri točke na mreži koje okružuju točku (x, y).

Funkcija

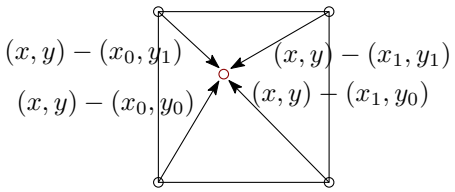
$$g(x_m, y_m) = (g_x, g_y), \quad x_m, y_m \in \mathbb{Z}$$

Svakoj točki na mreži dodjeljuje se pseudoslučajni normalizirani gradijent. To znači da iako gradijent djeluje nasumično, uvijek je isti za istu točku mreže. Također je važno da je dobra slučajna razdioba, tj. da svaki smjer ima jednaku vjerojatnost da bude odabran.



Slika 2.4: Četiri pseudoslučajna gradijenta pridružena točkama mreže.

Za svaku se točku mreže određuje i vektor prema točki (x, y) koji se dobiva oduzimanjem vektora točke mreže od (x, y). (slika 2.5)



Slika 2.5: Vektori iz točaka mreže prema točki (x, y).

Sada su definirani svi potrebni podaci za računanje vrijednosti funkcije šuma na koordinatama (x, y). Prvo se određuje faktor utjecaja svakog pseudoslučajnog gradijenta na konačni izlaz koji se računa kao težinski prosjek tih faktora.

Utjecaj svakog gradijenta računa se kao skalarni produkt gradijenta i vektora usmjerenog prema točki (x, y).

Vrijednosti tih produkata označavamo sa s, t, u i v, pri čemu je

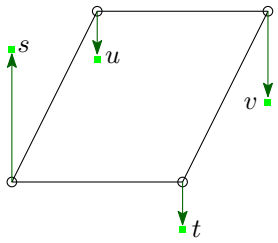
$s = g(x_0, y_0) \cdot ((x, y) - (x_0, y_0))$,

$t = g(x_1, y_0) \cdot ((x, y) - (x_1, y_0))$,

$u = g(x_0, y_1) \cdot ((x, y) - (x_0, y_1))$,

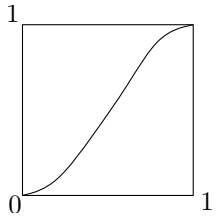
$v = g(x_1, y_1) \cdot ((x, y) - (x_1, y_1))$.

Na slici 2.6 u 3D prostoru prikazani su faktori utjecaja s, t, u i v. Dobivene vrijednosti potrebno je na neki način uprosječiti da bi se dobila konačna vrijednost funkcije šuma.



Slika 2.6: Utjecaji točaka mreže.

Da bi se dobili glađi prijelazi kod izlazne funkcije šuma potrebno je pojačati vrijednost većih faktora utjecaja. Za to se koristi funkcija $3p^2 - 2p^3$ koja ima oblik slova S te je pogodna za ovu primjenu (slika 2.7).



Slika 2.7: Funkcija koja dodatno izražava blizinu ulaza vrijednostima nula ili jedan.

Prvo se određuje vrijednost funkcije u točki $x - x_0$ te se dobiva faktor S_x. Nakon toga se određuje težinski prosjek vrijednosti s i t linearnom interpolacijom, s faktorom interpolacije s_x. Rezultat označavamo s a. Isto se ponavlja i za vrijednosti u i v, a rezultat označavamo s b. (slika 2.8)

Matematički:

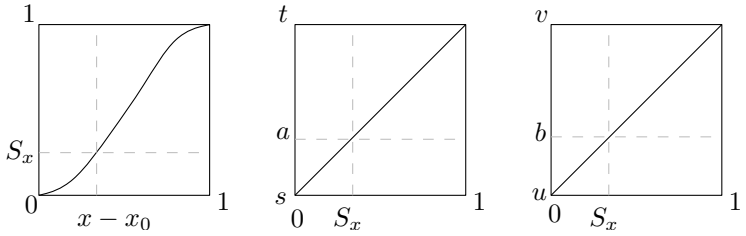
$S_x = 3(x - x_0)^2 - 2(x - x_0)^3$

$a = s + S_x(t - s)$

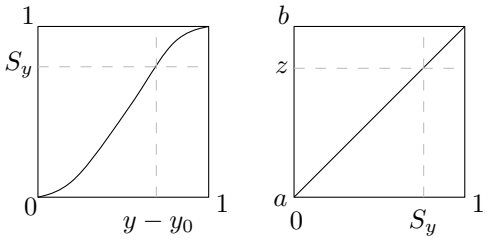
$b = u + S_x(v - u)$

Nakon toga se težinska vrijednost S_y dobiva evaluacijom funkcije u točki $y - y_0$ te se uzima težinska suma od a i b i dobiva se konačna izlazna vrijednost z. (slika 2.9)

Računalna složenost ovog algoritma ovisi o broju dimenzija. Potrebno je evaluirati 2^n točaka mreže i napraviti $2^{(n-1)}$ težinskih suma, čime se dobiva asimptotska složenost $O(2^n)$. To znači da algoritam postaje nepraktičan kod višedimenzionalnih prostora, ali je dovoljno brz za 2D i 3D prostore.



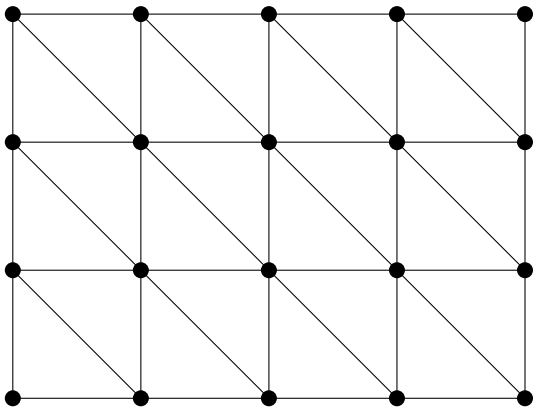
Slika 2.8: Nalaženje težinskih faktora linearnom interpolacijom.



Slika 2.9: Težinska suma dobivenih prosjeka daje konačni rezultat.

2.3. Generiranje poligona

Za prikaz u trodimenzionalnom prostoru potrebno je visinsku mapu pretvoriti u mrežu poligona. Vrhovi poligona sadrže veći broj vrijednosti, poput položaja u prostoru, smjera normale, koordinata tekstura, smjera tangente za preslikavanje normala i sl. Zbog toga za veće terene nije praktično spremati mreže vrhova i poligona, nego po potrebi iz visinske mape generirati mrežu vrhova. Svaka točka visinske mape postaje jedan vrh, pri čemu indeksi određuju njegov položaj na osima x i z, a vrijednost visinske mape položaj na osi y. Svaku od tih vrijednosti moguće je skalirati i na taj način dobiti više ili manje naglašene pojedine dimenzije (npr. povećan pomak na osi y za prenaglašeni prikaz planina). Dobivene točke povezuju se sa susjednima i čine pravokutne trokute čije su katete paralelne s osima visinske mape (sl. 2.10)

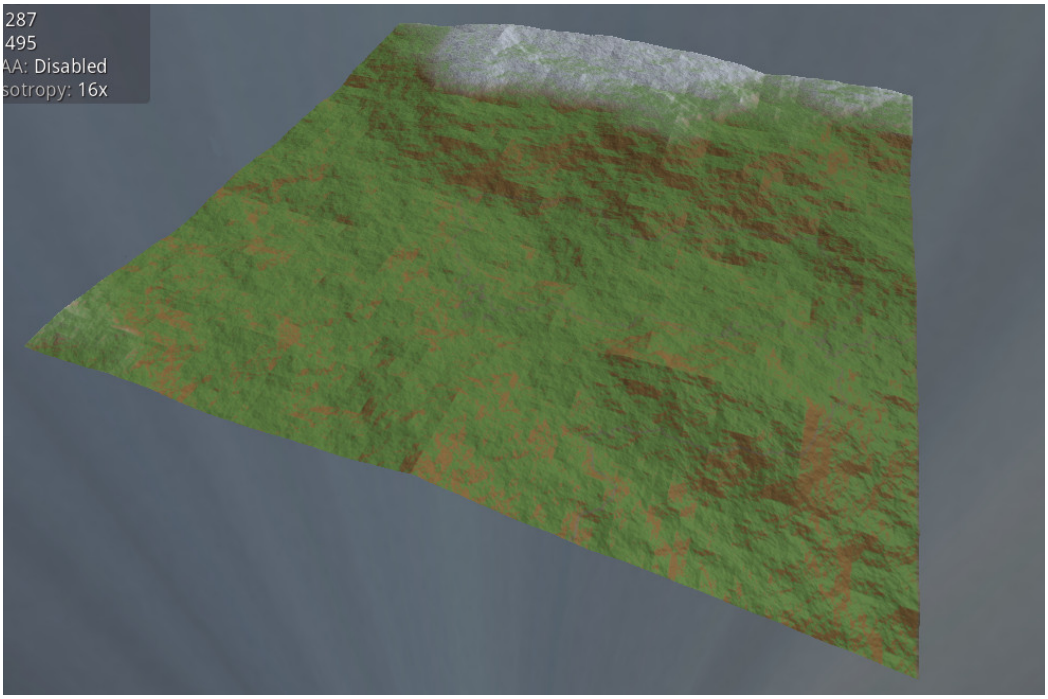


Slika 2.10: Prikaz povezivanja vrhova u poligone.

Zbog svoje veličine, nije dobro cijeli teren spremiti u jedan spremnik vrhova i svaki put ga cijelog iscrtavati. Veći se tereni dijele u odvojene blokove (komade, engl. *chunks*) koji se mogu odvojeno iscrtavati. Na taj način mogu se prije iscrtavanja odbaciti oni dijelovi terena koji su izvan vidnog polja (*view frustum culling*) te se omogućuje sortiranje pojedinih blokova prema udaljenosti od promatrača čime se izbjegava višestruko precrtavanje istih piksela.

2.4. Teksturiranje terena

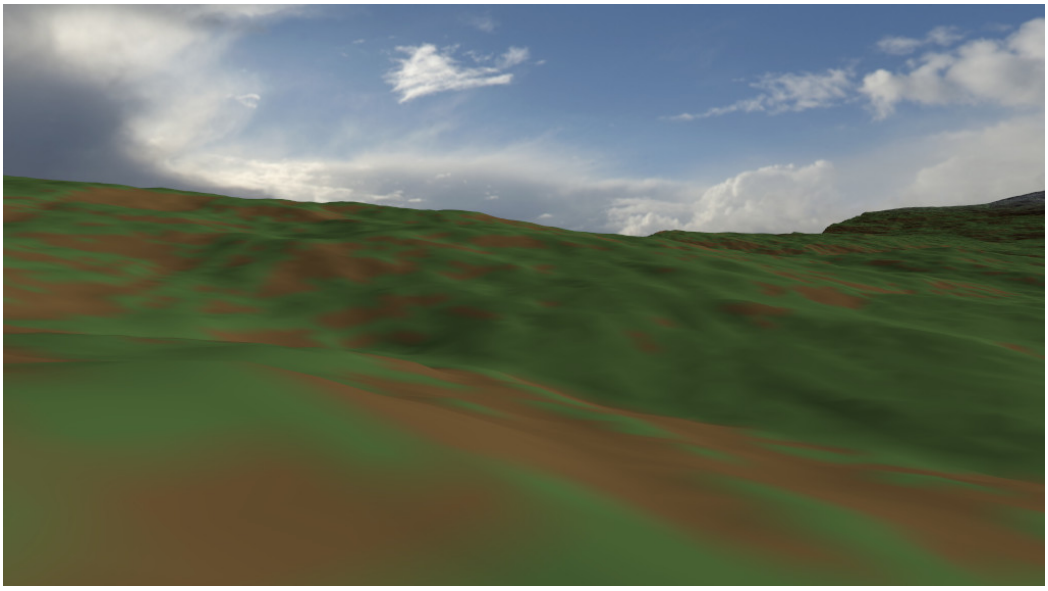
Uz podatke o visini određene točke, za vjeran prikaz terena potrebno je i odrediti materijal, odnosno boju pojedine točke. Boje točaka se isto spremju u obliku slike. Ta slika može biti drugačijih dimenzija od visinske mape (ovisno o željenoj razini detalja boje terena).



Slika 2.11: Obojan teren prikazan izdaleka.

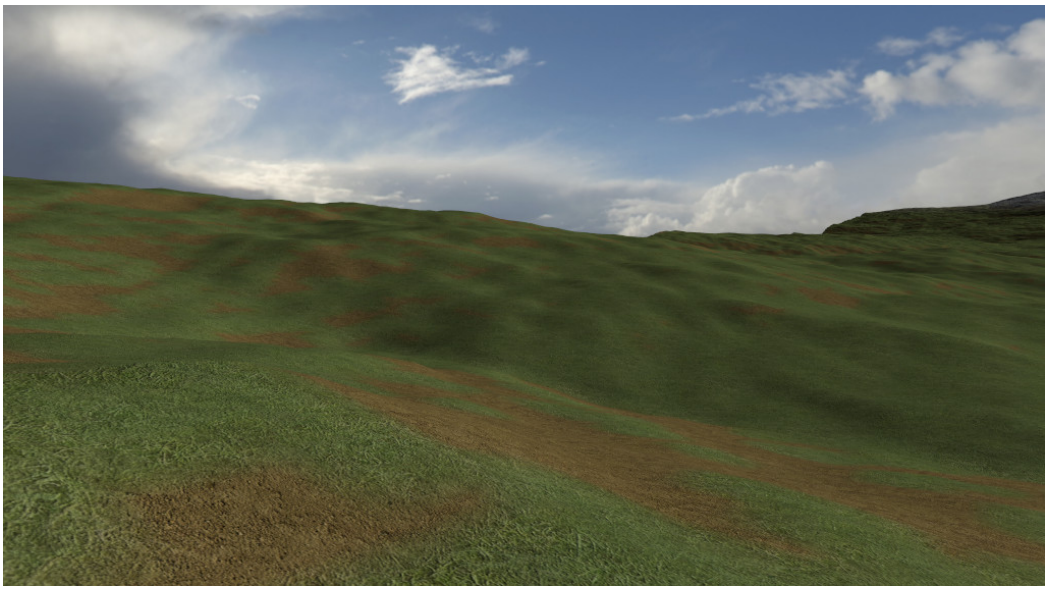
Na slici 2.11 može se vidjeti teren generiran visinskom mapom dimenzija 1024 x 1024, i s istom rezolucijom teksture za boju terena. Na slici 2.12 vidimo isti teren prikazan izbliza, pri čemu se može vidjeti nedstatak detalja i mutan izgled teksture. Ako pretpostavimo da jedna veličine terena iznosi jedan metar, a teren promatramo iz ljudske perspektive, pojedini pikseli teksture su preveliki za vjeran prikaz terena.

Povećanje dimenzija teksture nije moguće rješenje zbog memorijskih ograničenja.



Slika 2.12: Prikaz obojanog terena iz ljudske perspektive.

Čovjek koji stoji može na tlu ispod sebe vidjeti varijacije u boji veličine jednog milimetra, primjerice pojedine vlati trave ili komadiće zemlje. Takva bi razina detalja za teren dimenzija 1024 x 1024 metara zahtijevala teksturu boje rezolucije 1024000 x 1024000, što bi u RGB formatu zauzimalo 2,86 terabajta memorije. Jednostavnije rješenje je korišenje drugih tekstura, veće frekvencije koje se ne rastežu preko cijelog terena, nego se teren njima popločava.



Slika 2.13: Prikaz obojanog i popločenog terena iz ljudske perspektive.

Postoji veći broj metoda za određivanje kombinacije detaljnih tekstura za neki dio

terena. Za svaki se piksel detaljnim teksturama pridružuju se težinski faktori koji određuju koliki je udio svake teksture u konačnoj boji piksela. To se može napraviti unaprijed definiranim težinskim mapama, ali se može izračunati i dinamički, pomoću boje piksela niskofrekventne teksure terena. Za glatke prijelaze između boja teksturu boja terena potrebno je bilinearno filtrirati.

Svakoj se detaljnoj teksturi pridružuje jedna boja, primjerice zelena za travu, smeđa za tlo, i sl. Ta boja može se odrediti ručno, a može se i generirati kao npr. prosječna ili najčešća boja piksela teksture. Kod prikaza terena svaki piskel svake detaljne mape miješa se s faktorom koji je obrnuto proporcionalan udaljenosti boje teksture terena od zadane boje teksture. To znači da će teksture koje imaju boju sličnu boji terena biti biti vidljivije, poput teksture tla preko smeđeg terena. Bilinearno filtriranje teksture terena znači da će i prijelazi između detaljnih tekstura biti glatki, a mogu se i modificirati preko dodatne detaljne težinske mape kojom se kontrolira prijelaz jednog terena u drugi.

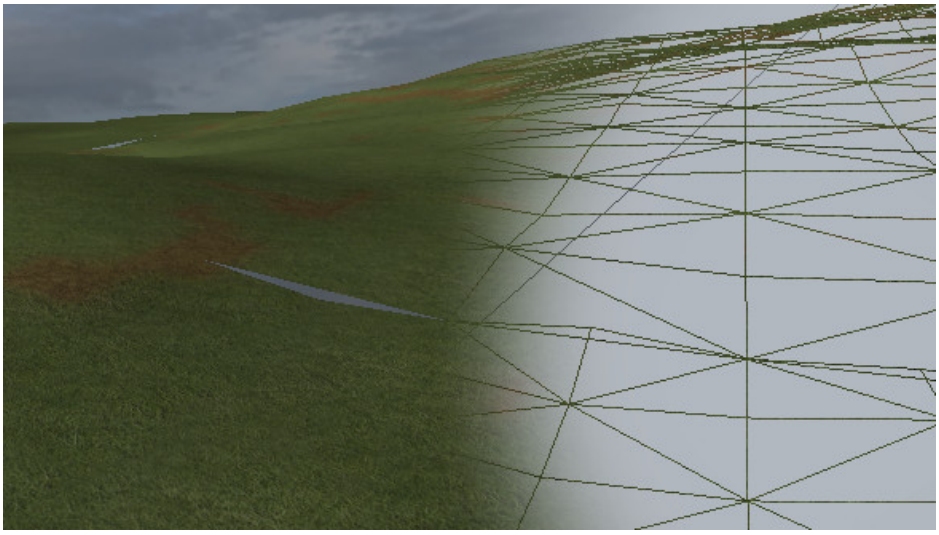
2.5. Razina detalja

U idealnom prikazu poligonalne scene svaki bi poligon imao podjednaku veličinu na ekranu. Zbog veličine terena udaljeni poligoni često pokrivaju samo nekoliko piksela, a oni najdalji mogu biti manji od piksela. Budući da je svaki vrh poligona potrebno transformirati prije rasterizacije, potrebno je izbjeći nepotrebne izračune za udaljene dijelove terena, ali zadržati detalje prikaza kod bližih dijelova terena. Jedna od najpopularnijih metoda za prikaz razina detalja terena naziva se *geomipmapping*, odnosno geometrijske mipmape (de Boer, 2000).

Kod teksturiranja, prva (tj. nulta) mipmapa je sama tekstura, dok svaka sljedeća razina ima upola manju rezoluciju. Ovisno o udaljenosti od kamere, odabire se razina mipmape koja se prikazuje umjesto teksture visoke rezolucije. Slična metoda može se primijeniti i kod prikaza blokova terena. Izvorni blok terena predstavlja prvu razinu, a sljedeću razinu dobivamo odabirom svakog drugog vrha u mreži trokuta. Na isti način dobivaju se i više razine geometrijske mipmape koje imaju upola manje detalja od prethodne, što ih čini prikladnim za učinkovit prikaz na većoj udaljenosti. Mipmape se mogu odrediti unaprijed i spremiti u memoriju tijekom generiranja samog terena.

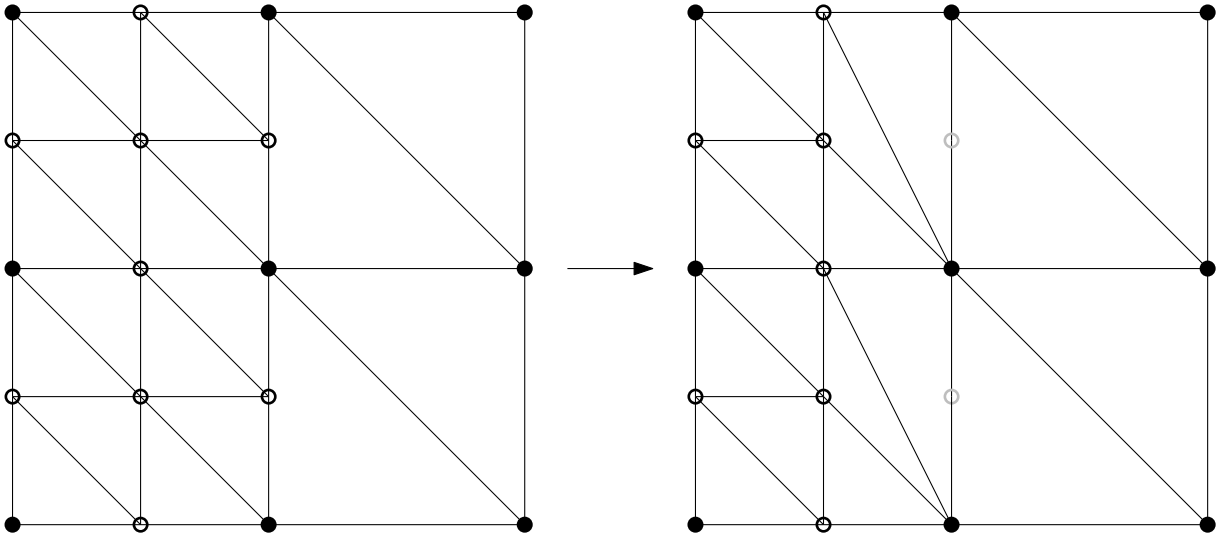
Kada susjedni blokovi imaju različitu razinu detalja pojavljuju se pukotine na prijelazima iz jednog bloka u drugi (slika 2.14).

Ovaj problem jednostavno se rješava premještanjem spojeva vrhova tako da se preskaču nespareni rubni vrhovi te se na taj način dobivaju jednaki rubovi na oba bloka



Slika 2.14: Prikaz pukotina na prijelazima između razina detalja.

(slika 2.15).

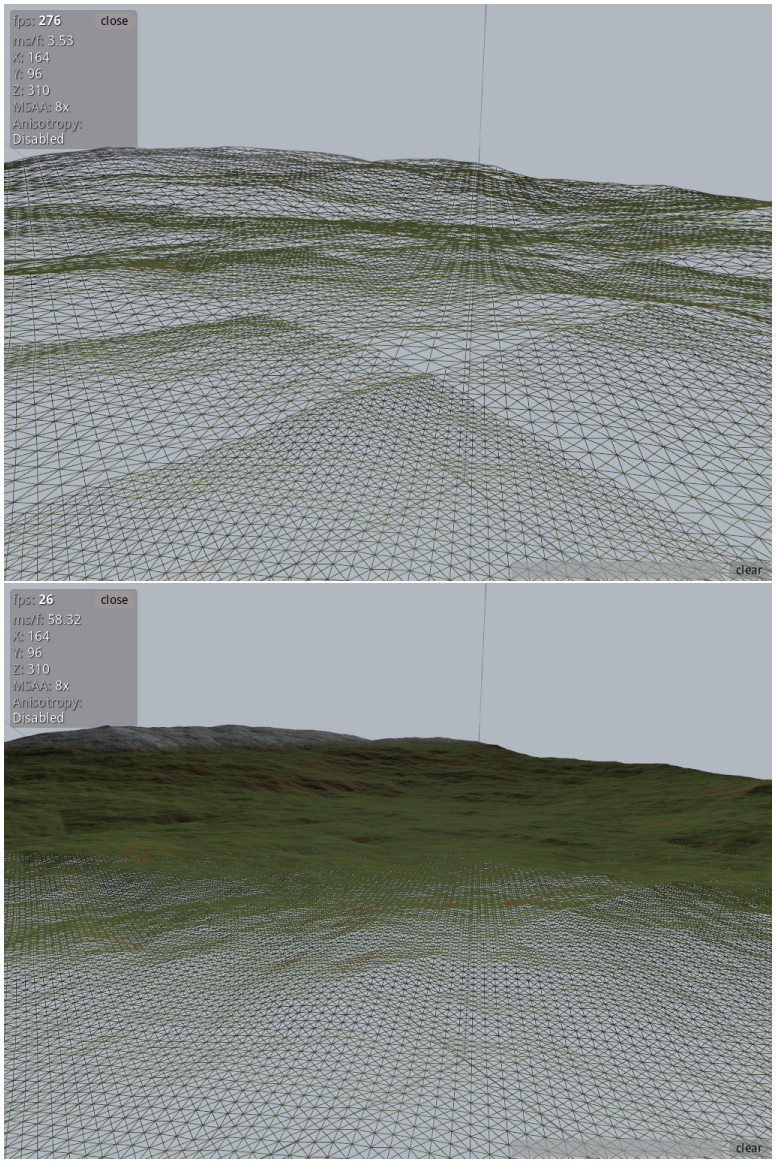


Slika 2.15: Uklanjanje pukotina na prijelazu između dvije različite razine detalja.

Za svaki blok terena potrebno je odabrati kojom će razinom detalja biti prikazan. Redni broj razine ovisi o udaljenosti od promatrača. Time se postiže ravnomjerniji raspored veličina trokuta, ali i manje vidljiv prijelaz zbog promjena visine točaka kod prebacivanja bloka iz jedne razine u drugu. Potrebno je odabrati udaljenosti koje su dovoljno velike da minimiziraju prividnu grešku kod prijelaza, ali i dovoljno male da ne iscrtavamo presitne poligone za udaljene blokove terena. Razlika u prikazu terena s uključenom i isključenom kontrolom razine detalja prikazana je na slici 2.16.

U ovom radu je za prikaz terena korištena je modificirana implementacija ge-

omipmapping algoritma opisana u poglavlju 4.1.



Slika 2.16: Žičani prikaz terena sa i bez korištenja kontrole razine detalja.

3. Tereni zasnovani na volumnim mapama

3.1. Volumna mapa

Volumna mapa, odnosno trodimenzionalno skalarno polje jest volumen u kojemu je svakoj prostornoj točki pridružena neka skalarna vrijednost. Može se izraziti kao preslikavanje

$$f : \mathbb{R}^3 \to \mathbb{R}.$$

Kod prikaza terena, glavna razlika terena zasnovanih na volumnim mapama u odnosu na terene zasnovane na visinskim mapama je mogućnost volumnih mapa da opišu prirodne pojave poput špilja, tunela, okomitih litica, jama, lukova i raznih stjenovitih formacija nastalih erozijom. Na slici 3.1 možemo vidjeti litice, špilju i luk koji se nemogu opisati visinskom mapom.

Razlog tome je što za razliku od visinkih mapa, volumne mape ne spremaju vrijednost visine točke. Visina je sada zadana implicitno pomoću indeksa koordinate y. Vrijednost koja se kod volumnih mapa pridružuje točkama u prostoru je gustoća. Pozitivna vrijednost gustoće znači da točka volumena prestavlja čvrstu tvar (kamen, tlo), a negativna predstavlja zrak.

Za generiranje volumne mape mogu se koristiti trodimenzionalne inačice istih algoritama kao i za generiranje visinske mape.

Takvi podaci mogu se prikazati izravno (metode prikaza volumena), a mogu se na temelju podataka generirati geometrijske primitive koje se potom prikazuju.

Slika 3.1: Litice, špilja i luk pored mjesta Étretat u Francuskoj koje se ne bi mogle prikazati pomoću visinskih mapa.

3.2. Praćenje zrake

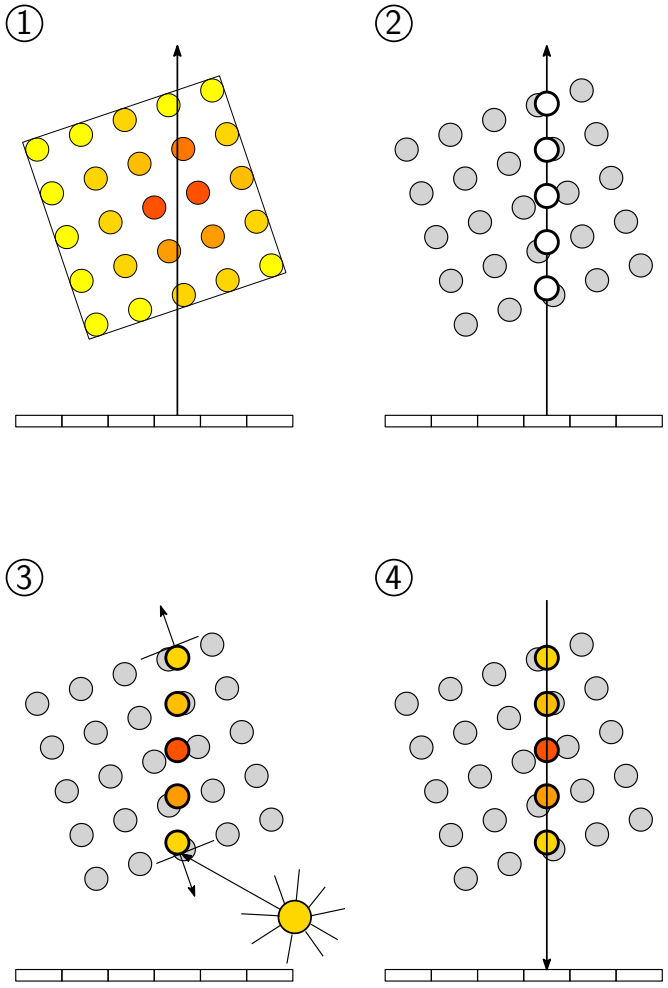
Postupak praćenja zrake kroz volumen je tehnika prikaza u prostoru slike. Brzina prikaza ovisna je o rezoluciji slike i veličini volumne mape, a ne o njezinoj geometrijskoj složenosti.

Algoritam se sastoji od četiri osnovna koraka koji su prikazani na slici 3.2.

1. Za svaki slikovni element konačne slike ispaljuje se zraka kroz volumen. Volumen se omeđuje kvadrom, a točke presijeka s omeđujućim kvadrom su točke u kojima zraka ulazi i izlazi iz volumena.

2. Na putu zrake kroz volumen odabiru se ispitne točke raspoređene u pravilnim razmacima. U pravilu, osi volumena nisu poravnate sa zrakom pa se ispitne točke uglavnom nalaze između točaka volumena. Zbog toga je potrebno interpolirati vrijednosti uzoraka susjednih točaka volumena, najčešće trilinearnom interpolacijom.

3. Za svaku ispitnu točku računa se gradijent vrijednosti osvjetljenja. Ti gradijenti predstavljaju orijentaciju lokalnih površina unutar volumena. Zatim se uzorci



Slika 3.2: Prikaz postupka praćenja zrake kroz volumen.

sjenčaju prema orijentaciji svoje površine i položaju izvora svjetla u sceni.

4. Nakon sjenčanja svih ispitnih točaka, vrši se njihova kompozicija te se dobiva konačna boja slikovnog elementa. Kompozicija se vrši u smjeru suprotnom od smjera zrake. Počinje se od uzorka najudaljenijeg od promatrača, a završava onim najbližim. To osigurava da prekriveni dijelovi volumena ne utječu na boju slikovnog elementa.

Ovisno o željenoj projekciji određuje se smjer svake zrake. Kod ortografske projekcije zrake putuju paralelno iz smjera promatrača, a kod perspektivne projekcije određuje se točka promatrača i "platno" koje se nalazi između promatrača i promatrane scene. Na platnu se nalazi mreža slikovnih elementata, a smjer je vektor od točke kamere prema točki slikovnog elementa na platnu. Širina vidnog polja tada ovisi o udaljenosti platna od promatrača.

3.3. Pokretne kocke

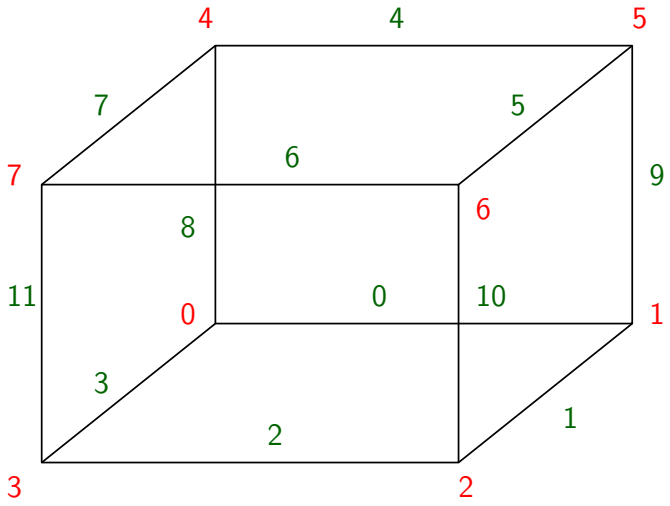
Pokretne kocke (engl. *Marching cubes*) su algoritam za stvaranje poligonalne mreže izopovršine trodimenzionalnog skalarnog polja. Algoritam su 1987. godine objavili Lorensen i Cline, a zamišljen je kao pomoć u medicinskoj dijagnostici.

Kao što je ranije navedeno, točke volumena u kojima je vrijednost gustoće negativna predstavljaju zrak, a one u kojima je pozitivna predstavljaju čvrstu tvar. Vrijednost gustoće za koju se određuje izopovršina je nula, granica čvrste tvari i zraka, odnosno površina terena.

U algoritmu se iterira kroz skalarno polje, uzimajući u obzir osam susjednih lokacija u svakom koraku koji čine kocku. Za svaku kocku određuju se poligoni potrebni za prikaz dijela izopovršine kroz tu kocku. Skup poligona dobivenih u svakoj kocki čini površinu objekta.

Glavni je problem nalaženje poligona koji najbolje predstavljaju izopovršinu u svakoj kocki. Izopovršina može sjeći neku kocku, a može i proći izvan nje. Svaki od osam vrhova kocke može se nalaziti iznad (manja vrijednost gustoće) ili ispod krivulje (veća vrijednost gustoće). Primjerice, ako kocka siječe ravno tlo donje četiri točke će biti ispod, a gornje četiri iznad krivulje. Ako je jedan vrh kocke iznad površine, a njemu susjedni vrh ispod – znači da površina siječe brid između ta dva vrha. Položaj na kojem površina siječe brid određuje se linearnom interpolacijom, a omjer udaljenosti vrhova od površine jednak je omjeru apsolutne vrijednosti razlike njihove vrijednosti od vrijednosti površine.

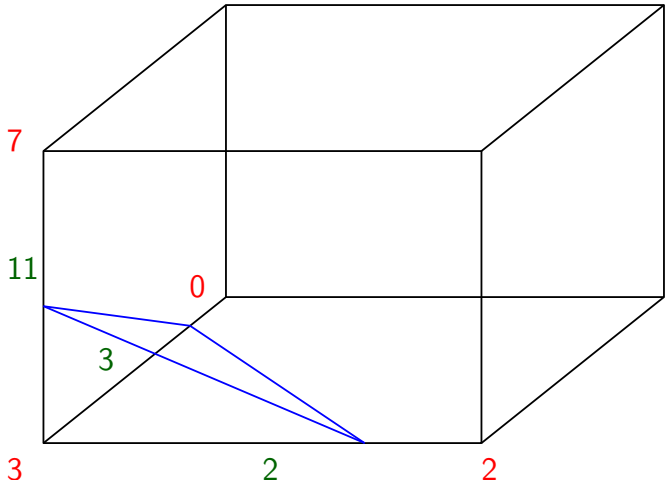
Indeksi vrhova i bridova u algoritmu prikazani su na slici 3.3.



Slika 3.3: Indeksi vrhova i bridova kocke.

Primjerice, ako je vrh 3 ispod površine, a svi drugi vrhovi su iznad vrijednosti

površine tada se stvara poligon koji siječe bridove 2, 3, i 11, a točan položaj vrhova poligona na bridovima ovisi o odnosima vrijednosti gustoće u parovima vrhova 3 i 2, 3 i 0 te 3 i 7 (slika 3.4).



Slika 3.4: Prikaz generiranja poligona ako je vrh 3 ispod, a ostali vrhovi iznad površine.

Glavni problem u radu algoritma je veliki broj mogućih slučajeva (256), kao i potreba da se definira konzistentne kombinacije poligona za svaki mogući slučaj, tako da poligoni susjednih kocki budu dobro povezani (bez pukotina).

Prvi dio algoritma koristi unaprijed generiranu tablicu bridova koja preslikava kombinaciju vrhova koji se nalaze ispod površine u niz bridova koje površina siječe. Slaže se 8 bitni indeks kod kojeg svaki bit odgovara jednom vrhu.

```
cubeindex = 0;
if (grid.val[0] < isolevel) cubeindex |= 1;
if (grid.val[1] < isolevel) cubeindex |= 2;
if (grid.val[2] < isolevel) cubeindex |= 4;
if (grid.val[3] < isolevel) cubeindex |= 8;
if (grid.val[4] < isolevel) cubeindex |= 16;
if (grid.val[5] < isolevel) cubeindex |= 32;
if (grid.val[6] < isolevel) cubeindex |= 64;
if (grid.val[7] < isolevel) cubeindex |= 128;
```

Preslikavanjem dobivenog indeksa pomoću tablice vrijednosti dobiva se 12 bitni broj u kojem svaki bit predstavlja jedan brid. Ako je vrijednost bita 0, znači da površina ne siječe brid, a ako je 1 znači da ga siječe. Ako površina ne siječe ni jedan brid, tablica vraća 0. To se događa kada su svi vrhovi iznad površine (indeks 0) ili ispod površine (indeks 0xff).

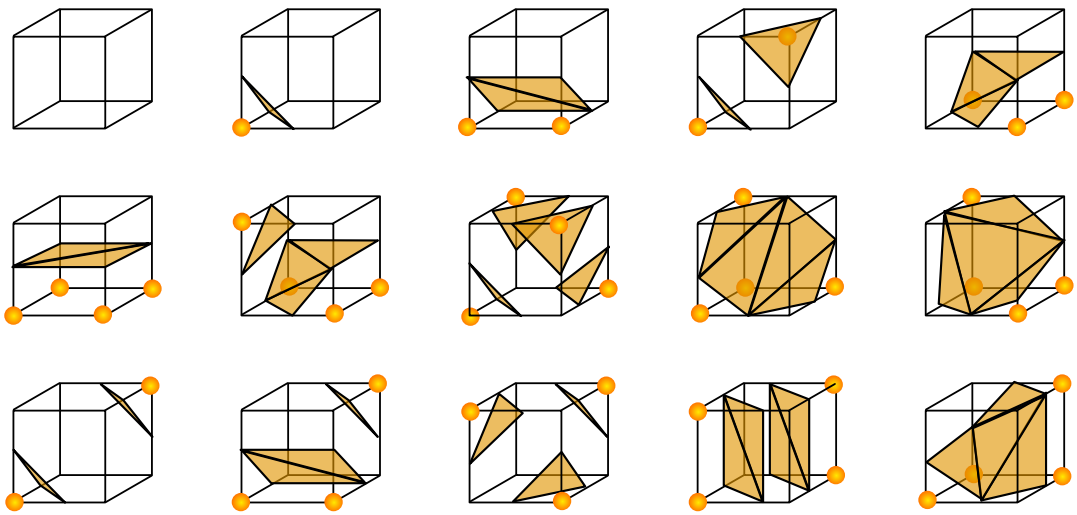
U primjeru gdje je samo vrh 3 ispod površine, indeks bi bio 0000 1000, odnosno 8. Tablica bridova za indeks 8 daje niz bitova 1000 0000 1100, što znači da površina siječe bridove 2, 3 i 11.

Točke u kojima površina siječe brid računaju se linearnom interpolacijom. Ako su T_1 i T_2 vrhovi brida, a V_1 i V_2 skalarne vrijednosti gustoće u vrhovima, onda se točka presjeka P dobiva kao

$$P = T_1 + (V_{izo} - V_1)\frac{T_2 - T_1}{V_2 - V_1}.$$

U drugom dijelu algoritma određuju se ispravne plohe (poligoni) iz točaka u kojima izopovršina siječe bridove kocke. Definira se nova tablica – tablica trokuta koja koristi isti indeks kao i tablica bridova, ali vraća kombinacije bridova (tj. točaka na bridovima) koji čine vrhove trokuta.

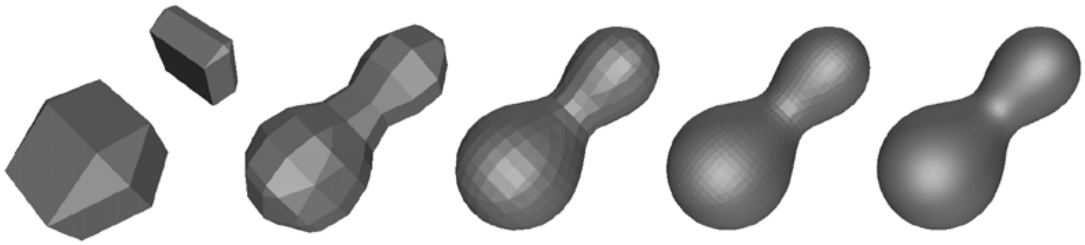
Ako zanemarimo dvije spomenute trivijalne kombinacije (svih osam vrhova iznad ili ispod površine), preostale 254 kombinacije možemo svesti na 14 posebnih kombinacija ako uzmemo u obzir simetričnost. Prazna kocka i 14 posebnih slučajeva prikazani su na slici 3.5.



Slika 3.5: Moguće kombinacije generiranih poligona.

Ovisno o željenoj razini detalja prikaza i računalnim mogućnostima koje imamo na raspolaganju definira se veličina kocki, odnosno veličina same volumne mape (slika 3.6).

Posljednja stvar koju treba izračunati je normala svakog vrha. Normala se računa parcijalnom derivacijom funkcije gustoće u smjeru x, y i z osi i normalizacijom dobivenog vektora. Jednostavan način računanja je uzorkovanje volumne mape šest puta. Za dobivanje promjene u smjeru x osi, uzimaju se uzoraci susjednih elemenata mape u



Veličina kocke = 10
70 poligona

Veličina kocke = 5
220 poligona

Veličina kocke = 2
1700 poligona

Veličina kocke = 1
6800 poligona

Veličina kocke = 0.5
27000 poligona

Slika 3.6: Ovisnost vjernosti prikaza izopovršine o veličini volumne mape.

+x i -x smjeru. Računa se njihova razlika i tako se dobiva promjena u smjeru x osi. Na isti način se računa promjena u smjeru y i y osi. Tri dobivene vrijednosti čine vektor koji se potom normalizira. Na taj se način dobivaju vrlo kvalitetne i glatke normale koje se koriste za osvjetljenje.

Razina detalja može se implementirati modifikacijom geomipmapping algoritma tako da se programu za generiranje poligona šalju umanjene volumne mape i spremanjem dobivenih poligona. Zbog nepravilnih oblika dobivenih pomoću algoritma pokretnih kocki nije dovoljno samo promijeniti indekse kao kod terena zasnovanog na visinskim mapama, već je potrebno spremati sve generirane vrhove.

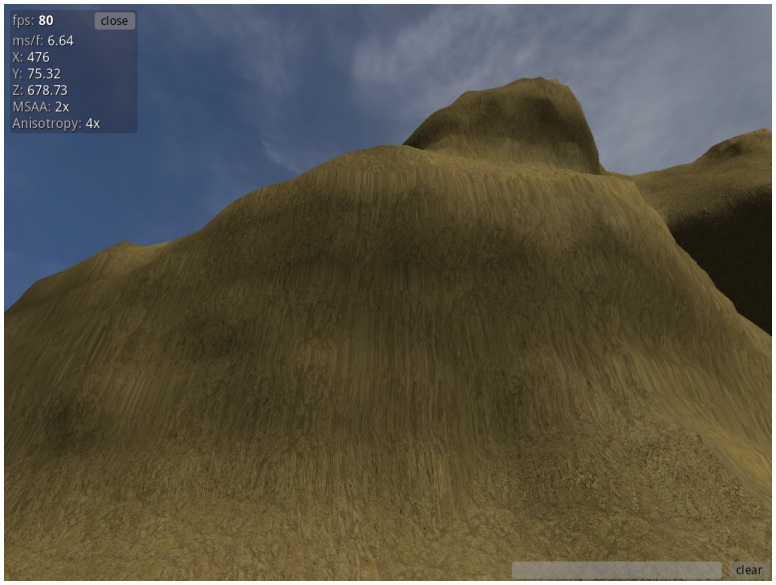
3.4. Teksturiranje terena

Jedan od problema kod proceduralnog generiranja terena ili drugih proizvoljnih oblika je teksturiranje, tj. stvaranje koordinati tekstura. Potrebno je preslikati teksture na poligone s minimalnim izobličenjem. Kod terena zasnovanih na visinskim mapama koordinate teksture ovisile su o položaju točke na xz osi. To znači da je tekstura projcirana odozgo, što izgleda dobro ukoliko nema vrlo strmih dijelova terena. Strme dijelove terena je stoga poželjno izbjegavati kod takvog načina teksturiranja.

Međutim, volumni tereni mogu imati brojne okomite površine pa takav način teksturiranja nije zadovoljavajuć (slika 3.7).

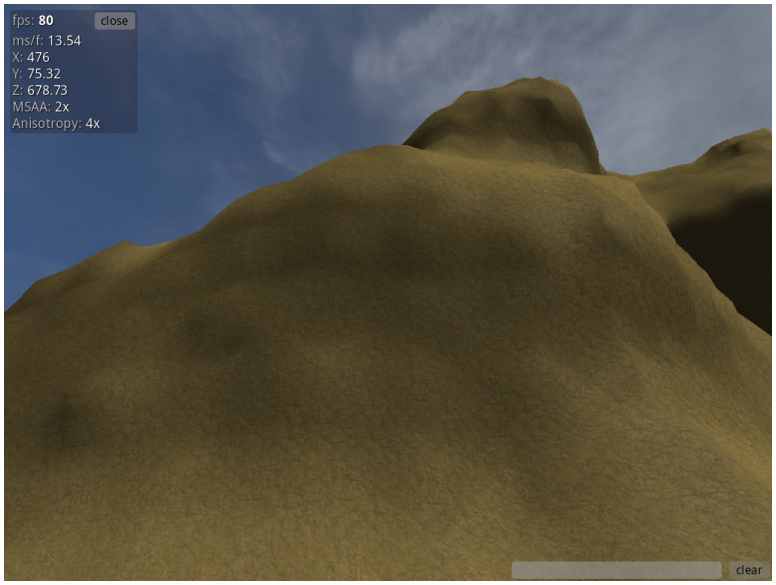
Jedno moguće rješenje je korištenje volumne teksture za prikaz volumnog terena. Svakoj točki odredi se trojka UVW koordinati za teksturu prema njezinom položaju u prostoru. Ukoliko se koristi veći broj tekstura ovo rješenje zahtijeva velike količine memorije. Jedna RGBA tekstura dimenzija 512 piksela visine, širine i dubine zauzima $4\,\mathrm{B} \cdot 512^3 = 512\,\mathrm{MB}$.

Drugo rješenje je troplanarno teksturiranje koje omogućuje prikazivanje dvodimenzionalnih tekstura preko trodimenzionalnih koordinata. Kao koordinata za tek-



Slika 3.7: Rastezanje teksture projcirane odozgo na strmoj litici.

sturu koristi se trodimenzionalni položaj točke u prostoru xyz koja se dijeli na tri para dvodimenzionalnih koordinati xy, yz i xz. Za teksturiranje svake točke odabire se ona projekcija teksture kod koje dolazi do najmanje rastezanja, a na prijelazima se radi miješanje između tekstura (slika 3.8).

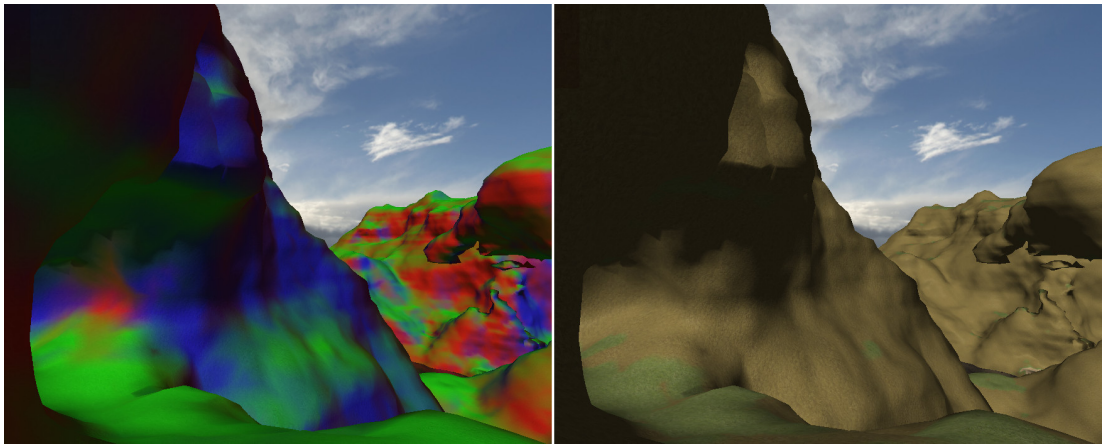


Slika 3.8: Rastezanja nema kod troplanarne projekcije teksture.

Za odabir kombinacije projekcija u nekoj točki koristi se njezina normala. Primjerice, za prikaz točke čija je normala usmjerena uglavnom u $+x$ ili $-x$ smjeru koristila bi se yz projekcija.

Da bi se spriječilo pretjerano miješanje i zamućivanje tekstura dominantna kompo-

nenta normale se pojačava, a ostale smanjuju. Na slici 3.9 crvenom bojom označena su područja s dominantnim x smjerom normale, a zelenom i plavom ona u kojima dominira y ili z smjer.



Slika 3.9: Prikaz faktora pojedinih osi za troplanarno teksturiranje.

4. Implementacija

Za implementaciju rada korišten je programski jezik C++, grafička biblioteka OpenGL, jezik za sjenčanje GLSL, biblioteka glm koja u jeziku C++ implementira matematičke funkcije i tipove podataka iz jezika GLSL, biblioteka assimp za učitavanje 3D modela, biblioteka za učitavanje tekstura lodepng te sustav SFML za upravljanje prozorima i unosom.

Zbog korištenja programa za obradu geometrije za generiranje poligona grafičko sklopovlje mora podžavati najmanje OpenGL 3.

Napravljen je sustav za generiranje terena zasnovanih na visinskim i volumnim mapama, njihovo prikazivanje i teksturiranje. Napravljeni su i dodatni elementi koji poboljšavaju vizualni dojam terena, poput vode te cvijeća i raslinja.

4.1. Prikaz terena zasnovanih na visinskim mapama

Sustav za prikaz terena zasnovanih na visinskim mapama sastoji se od:

- visinske mape i funkcija za generiranje visinske mape i boje terena,
- klase za prikaz pojedinog bloka terena,
- klase koja se bavi prikazom cijelog terena.

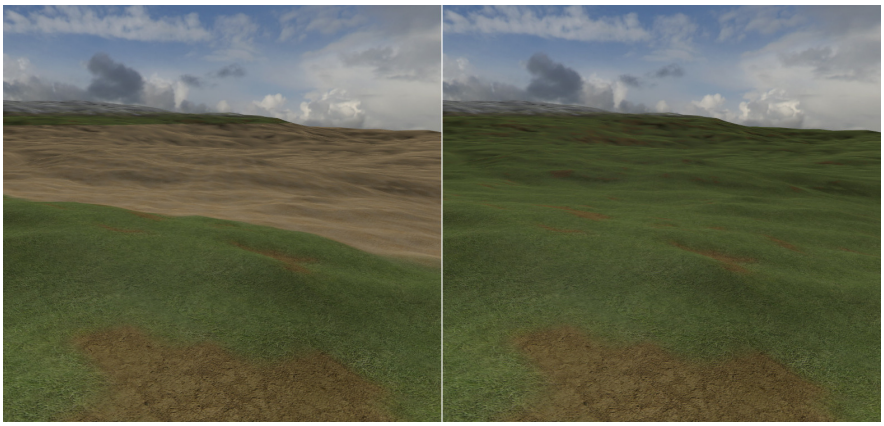
Visinska mapa generira se pomoću algoritma dijamant-kvadrat, a boja terena generira se na grafičkoj kartici čime se omogućuju veće promjene u boji terena u stvarnom vremenu.

Generiranje boje terena ostvareno je crtanjem u teksturu, odnosno u spremnik okvira, umjesto na ekran. Funkcija je implementirana kao program za sjenčanje fragmenata. Predmet koji se crta je četverokut koji pokriva cijeli ekran, tj. spremnik okvira. Programu se prosljeđuje već izračunata visinska mapa u obliku teksture. Vrijednosti spremljene u visinskoj mapi mogu pokrivati vrlo širok raspon visina, a razlučivost od $1/256$ koju omogućuje jedan kanal teksture nije dovoljna za vjeran prikaz terena. Zbog

toga se vrijednost u visinskoj mapi sprema u sva četiri kanala RGBA teksture što daje željenu razlučivost i raspon vrijednosti visinske mape.

Algoritam za generiranje boje dobiven je eksperimentiranjem sa željom da rezultat bude vizualno zadovoljavajuć, ali i jednostavan za implementaciju. Složeniji algoritmi dali bi bolje i uvjerljivije rezultate.

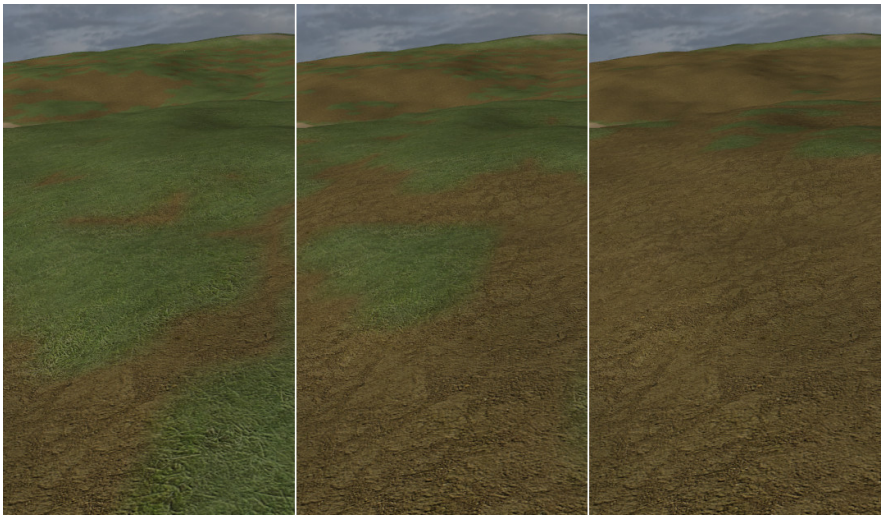
Boja svake točke ovisi o dva parametra – njezinoj visini i kutu koji njezina normala zatvara sa osi y. Boje koje ovise o visini su boja pijeska (ispod određene visine) i boja snijega (češća iznad određene visine). Razlike u izgledu terena s promjenom parametra visine za boju pijeska mogu se vidjeti na slici 4.1. Boje ovisne o normali su boja zemlje i boja trave. Ako je kut između osi y i normale veći od zadanog praga znači da je ta površina relativno ravna i prikazuje se u boji trave, ako je kut manji točka se prikazuje u boji zemlje. Razlike u izgledu terena kod promjene praga za boju trave prikazan je na slici 4.2.



Slika 4.1: Promjena visine praga za pijesak kod generiranja teksture.

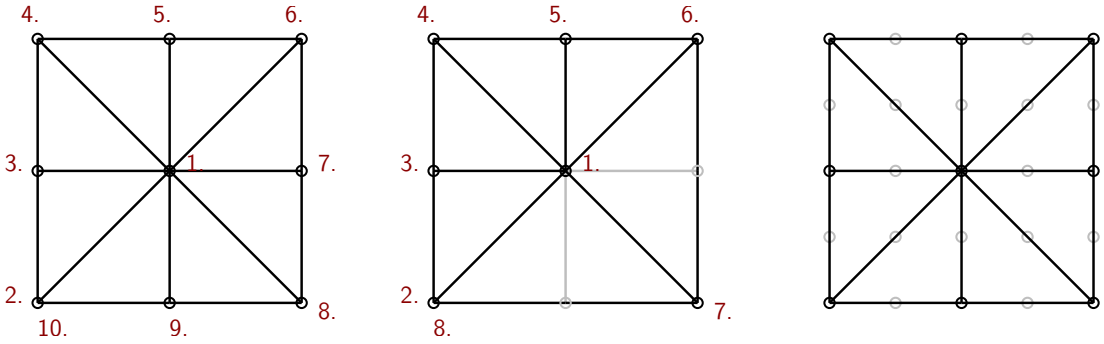
Blok terena ima funkcije za generiranje vrhova, indeksa i crtanje terena. Kod generiranja vrhova bloku se proslijeđuje dio visinske mape koji želimo u njemu prikazati. Svakom vrhu pridružuje se visina, a prema njegovom položaju određuju se i koordinate tekstura za mapu boja i teksture detalja.

Dobiveni vrhovi još nisu povezani u poligone. Povezuju se pomoću niza indeksa koji se stvaraju u posebnoj funkciji za generiranje indeksa. Indeksi se generiraju tako da vrhove povezuju u lepezu trokuta. Oblik lepeze trokuta odabran je zbog jednostavnosti implementacije prijelaza između razina detalja. Za svaku lepezu odabire se devet točaka, jedna središnja i osam okolnih. Točke se povezuju u lepezu kako je prikazano na slici 4.3. Na sredini slike prikazana je lepeza s preskočenim središnjim točkama nekih stranica. To znači da će na tim stranicama susjedni blok biti manje razine detalja, a preskakanje tih vrhova (koji ne postoje kod niže razine de-



Slika 4.2: Promjena faktora koji određuje količinu trave.

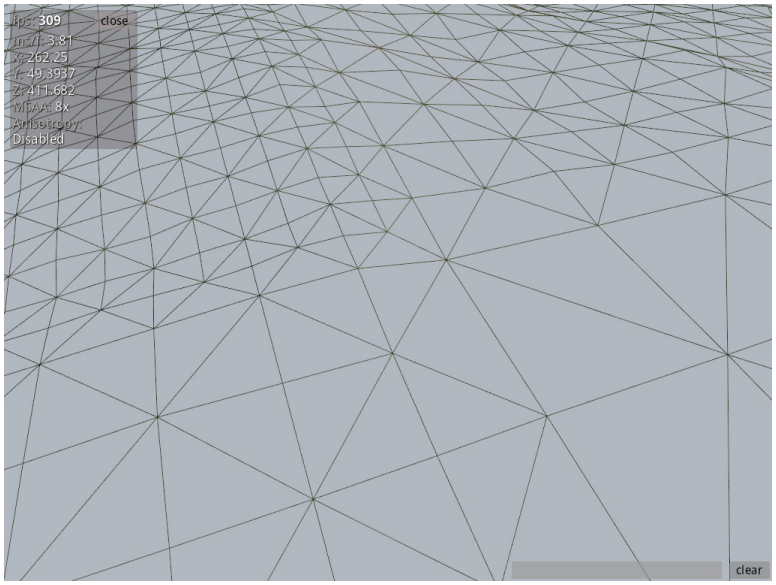
talja) omogućuje prikaz terena bez pukotina. Nakon što se obiđe svih osam okolnih vrhova (i preskoče oni koji se nalaze uz blok manje razine detalja) povezuje se zadnji vrh sa drugim i središnjim tako da se lepeza zaokruži.



Slika 4.3: Vrhovi visinske mape povezuju se u lepeze trokuta.

Kod generiranja indeksa za niže razine detalja preskaču se susjedne i povećava se površina lepeze. Kod nulte razine detalja (cijela mreža) indeksiraju se sve točke visinske mape. U prvoj razini preskače se svaki drugi stupac i redak točaka (na slici 4.3 desno), a u svakoj sljedećoj razini se preskače svaki drugi stupac i redak prethodne razine. Za svaku razinu detalja, osim osnovne mape indeksa potrebno je unaprijed generirati skup indeksa za blokove kojima neki od susjeda ima nižu razinu detalja od njih. Budući da svaki blok ima najviše četiri susjeda i da svaki susjed može i ne mora imati nižu razinu detalja, potrebno je generirati $2^4 = 16$ skupova indeksa za svaku razinu detalja. S obzirom na to da svi blokovi mogu dijeliti isti skup indeksa, dovoljno ih je samo jednom generirati na početku izvođenja programa.

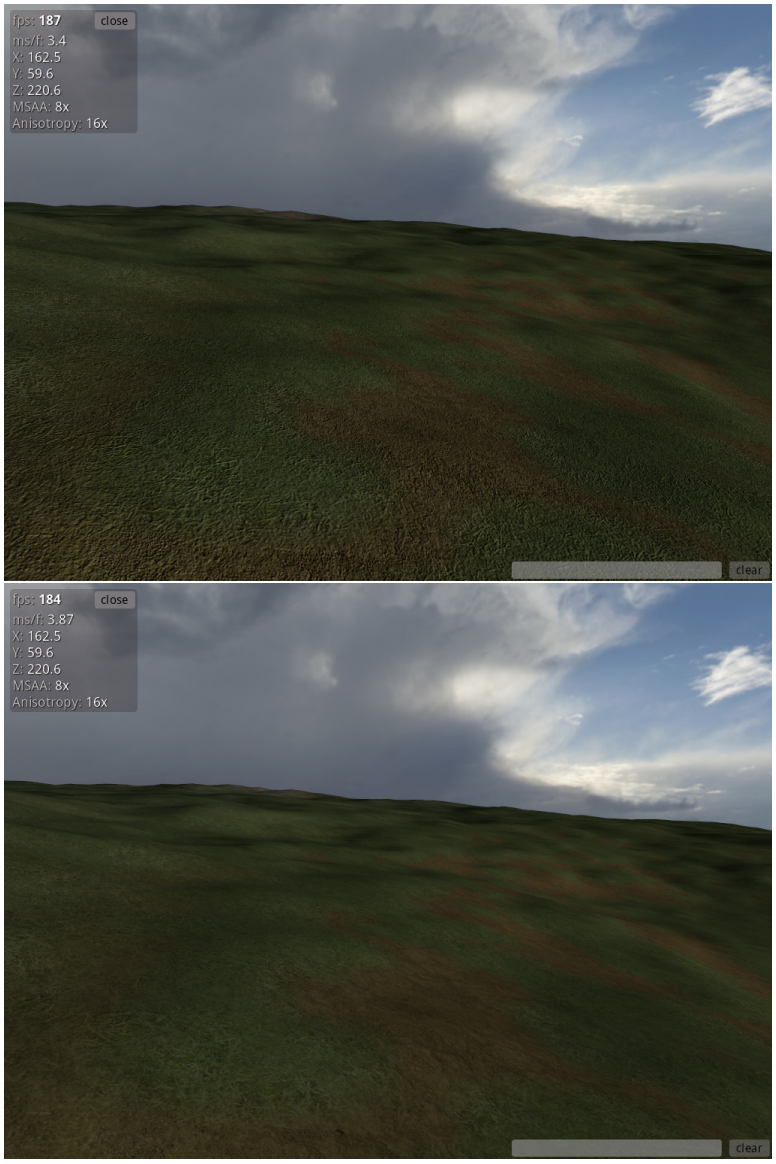
Teren predstavlja skup blokova terena i služi za prikazivanje svih blokova terena. Postupak prikazivanja odvija se u dva koraka. U prvom koraku prolazi se kroz sve blokove i prema njihovoj udaljenosti od kamere određuje se razina detalja kojom će biti prikazani. Nakon što se odrede razine detalja za sve blokove terena kreće korak crtanja u kojem se prvo gleda koji susjedni blokovi imaju nižu razinu detalja i prema tome se određuje skup indeksa koji se koristi u funkciji za crtanje bloka (slika 4.4). Blokovi se kod crtanja prvo transformiraju matricom terena, a nakon toga svojom matricom.



Slika 4.4: Prikaz granica između blokova s različitom razinom detalja.

Za crtanje koristi se metoda prikaza detaljnih tekstura opisana u poglavlju 2.4, s tim da je svakoj detaljnoj teksturi pridružena i odgovarajuća mapa normala čime se postiže puno prirodniji izgled terena. Razlika u prikazu terena s korištenjem i bez korištenja mape normala prikazana je na slici 4.5

Teren također ima funkcije za određivanje boje i visine terena za određenu točku u prostoru. Te funkcije se koriste za detekciju kolizije i određivanje materijala terena, što bi se moglo koristiti za određivanje zvuka koji proizvode koraci na tom dijelu terena. Te funkcije su implelentirane na razini terena, a ne na razini visinske mape jer teren ima vlastitu matricu transformacije koja određuje njegov položaj kao i skaliranje svake od osi terena. Te funkcije prvo trasformiraju zadane koordinate u lokalni prostor terena i zatim u prostor visinske mape. Vrijednosti boje i visine između točaka visinske mape određuju se bilinearnom interpolacijom.



Slika 4.5: Usporedba prikaza s preslikavanjem normala (gore) i bez preslikavanja normala (dolje).

4.2. Prikaz terena zasnovanih na volumnim mapama

Sustav za prikaz volumnih terena sastoji se od volumne mape, klase za prikaz pojedinog bloka terena i klase koja prikazuje cjelokupan teren.

Volumna mapa nije u cijelosti spremljena u memoriji, već se pomoću funkcija volumne mape stvaraju male volumne mape pojedinih blokova što bi moglo omogućiti beskonačno velike terene učitavanjem novih blokova u okolici promatrača i odbacivanjem starih, udaljenih.

Za generiranje točaka volumne mape koristi se Perlinov šum. Za svaku točku uzima

se sedam uzoraka šuma, različitih frekvencija i amplituda, tj. sedam oktava šuma, što omogućuje da teren ne izgleda monotono i da ima većih i manjih značajki. Konačna vrijednost gustoće sprema se u jedan oktet, kao jedna od 256 vrijednosti između 0 i 1. Vrijednosti gustoće ispod 0.5 predstavljaju zrak, a gustoće iznad 0.5 predstavljaju čvrstu tvar.

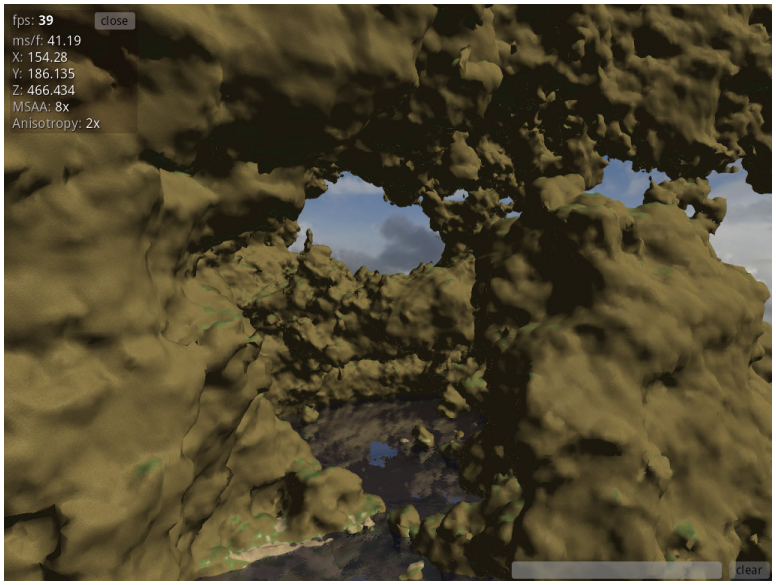
Radi bržeg generiranja blokova terena unaprijed se generira trodimenzionalna mapa Perlinovog šuma koja se kod generiranja blokova uzorkuje, umjesto računanja Perlinovog šuma za svaku točku svakog bloka terena. Posljedica toga je moguć malo monotoniji izgled terena (iako se to može izbjeći transformacijama koordinati prije uzorkovanja mape šuma) i stepeničast teren ako se koristi mapa šuma premale rezolucije. Prednost je veliko povećanje brzine generiranja terena. Za teren dimenzija 17x3x17 blokova od kojih svaki ima dimenzije 33x33x33 trajanje generiranja terena je s 65 sekunde za izravno uzorkovanje palo na 12 sekundi za uzorkovanje unaprijed generirane mape šuma. Vrijeme generiranje mape je zanemarivo u odnosu na vrijeme generiranja cjelokupnog terena jer se funkcija za generiranje Perlinovog šuma zove samo jednom za svaku točku mape, dok se za svaku točku terena zove sedam puta (za svaku oktavu šuma).

Daljnja prednost korištenja unaprijed generiranja mape šuma je biti mogućnost generiranja volumnih mapa na grafičkom sklopovlju crtanjem u trodimenzionalnu teksturu, što bi još više ubrzalo generiranje volumnih mapa.

Teren generiran na ovaj način vidljiv je na slici 4.6. Na slici se može vidjeti da izgleda vrlo strmo i neprirodno. Da bi teren izgledao više kao teren potrebno je uvesti ovisnost gustoće točke o nekom parametru, primjerice visini točke. Prije uzorkovanja početna vrijednost gustoće može se postaviti na neku početnu vrijednost i na taj način i prije uzorkovanja dati nekakav oblik terenu. Na slici 4.7 prikazan je teren generiran s različitim početnim vrijednostima gustoće točaka.

Na gornjoj slici početna je gustoća $1.0 - 2 * y$ (y se oduzima jer želimo da više točke imaju manju gustoću). Na srednjoj slici početna je gustoća $1.0 - y$, a na donjoj $1.0 - \sqrt{y}$. Možemo vidjeti da se jednostavnom promjenom početne gustoće mogu dobiti vrlo raznoliki tereni.

Najvažnije funkcije bloka terena su funkcije za generiranje mreže poligona i crtanje. Mreža poligona generira se pomoću algoritma pokretnih kocki koji je implementiran kao program za sjenčanje i izvodi se paralelno pomoću grafičkog sklopovlja. Primitiva koja se "crta" u tom programu je skup točaka koje predstavljaju koordinate tekstura volumne mape. Program za obradu vrhova samo prosljeđuje vrijednost koordinate teksture programu za obradu geometrije. Program za obradu geometrije postavl-



Slika 4.6: Izgled volumnog terena kad se ne uzme u obzir visina točke.

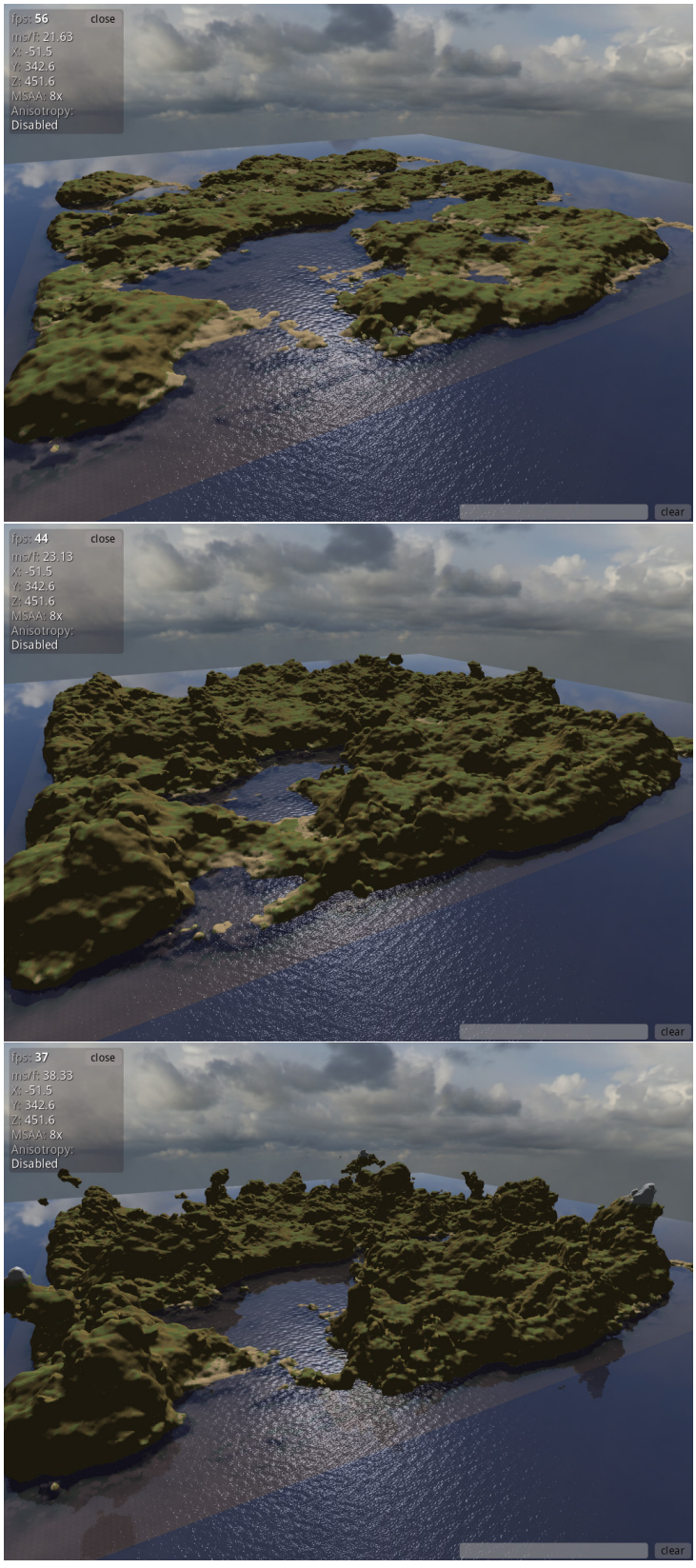
jen je da prima točku na ulazu i generira niz trokuta na izlazu. Svako izvođenje programa generira jednu kocku poligona, a cijelo crtanje generira čitav blok terena.

Generirani poligoni spremaju se u spremnik vrhova pomoću *transform feedbacka* koji omogućuje spremanje proizvoljnih podataka iz programa za sjenčanje u blok memorije. To znači da je moguće pisanje općenitijih programa za sjenčanje koji ne moraju imati veze s crtanjem. Prije izvođenja programa za generiranje poligona OpenGL-u se daje naredba za odbacivanje rasterizacije, što znači da nam nije potreban program za sjenčanje fragmenata.

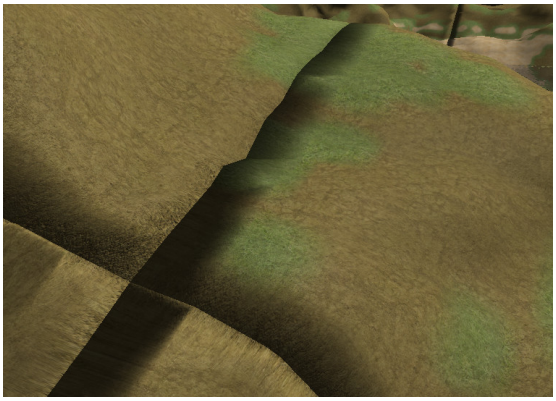
U programu se pomoću algoritma pokretnih kocki generiraju koordinate vrhova, koordinate tekstura i normale vrhova. Svaki blok terena ima svoju volumnu mapu, koja se koristi kao ulaz programa za generiranje poligona, a za generiranje normala potrebno je znati vrijednosti gustoće susjednih točaka da bi se mogao odrediti gradijent. Ako se u njihovoj volumnoj mapi nalaze samo one točke koje su poligonizirane, na prijelazima između blokova dolazi do grešaka u prikazu normala kao na slici 4.8.

Da bi se to izbjeglo svaki blok terena ima volumnu mapu koja je malo veća od terena koji prikazuje. Sa svih šest strana volumne mape dodaje se granica s podacima o vrijednostima gustoće u okolnim točkama. Ako prikazujemo blok terena dimenzija d_t u svakoj osi i želimo granicu debljine d_g, tada svaki blok terena ima volumnu mapu dimenzija $d_t + 2d_g$. Debljina granice množi se sa 2 jer se granica dodaje i s pozitivne i negativne strane osi. Da bi se izbjeglo crtanje tih graničnih područja, funkciji crtanja šaljemo samo koordinate središnjih ${d_g}^3$ točaka volumena.

Kod crtanja bloka koriste se obje vrste teksturiranja spomenute u poglavlju 3.4.



Slika 4.7: Utjecaj parametra visine točke na generiranje volumnog terena.



Slika 4.8: Greška u računanju normala na prijelazu između blokova terena.

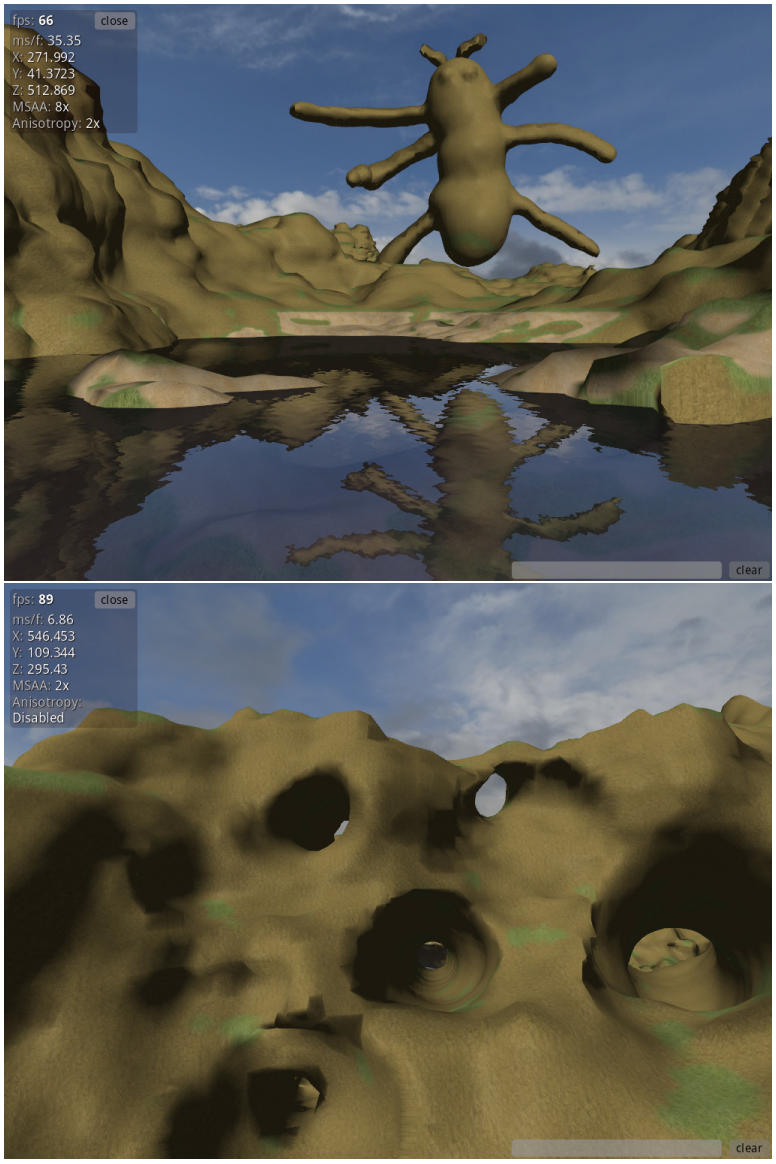
Za prikaz mape boja koristi se trodimenzionalna tekstura iste rezolucije kao i mapa gustoće. Boja je spremljena kao RGB vrijednost. Ušteda prostora mogla bi se ostvariti indeksiranjem boja i ručnom interpolacijom u programu za sjenčanjne fragmenata. Teksture detalja spremljene su kao 2D teksture i koristi se metoda troplanarne projekcije.

Kod klase za upravljanje cjelokupnim terenom zanimljive su funkcije za uređivanje terena. Budući da se mreža poligona generira pomoću grafičkog sklopovlja, moguće je generirati tisuće poligona u djeliću sekunde. To znači da je moguće u stvarnom vremenu promijeniti vrijednosti gustoća u volumnim mapama blokova i ponovno generirati te blokove. Moguće je osmisliti različite funkcije koje bi prema želji korisnika generirale različite oblike, a na slici 4.9 prikazan je rezultat korištenja implementiranih metoda koje linearno povećavaju ili smanjuju gustoću ovisno o radijusu unutar definirane sfere i na taj način omogućava "crtanje" terena u stvarnom vremenu.

4.3. Prikaz detalja

Za vjerniji prikaz terena nije dovoljno prikazati samo teren, nego je protrebno prikazati i druge elemente prirodnog okoliša poput neba, vode, drveća ili visoke trave. U ovom radu implementiran je prikaz neba u obliku kocke na koju je nalijepljeno šest tekstura. Drveće i drugi veći objekti mogu se ručno postaviti na scenu, a za prikaz raslinja definiran je poseban sustav.

Raslinje se sastoji od jednostavnog poligonalnog objekta (s vrlo malim brojem poligona) i teksture s definiranom mapom prozirnosti. Kod crtanja se odbacuju svi fragmenti čija je prozirnost manja od neke zadane granice, a ostali se crtaju potpuno neprozirno. Na taj način je omogućeno korištenje spremnika dubine za crtanje objekata



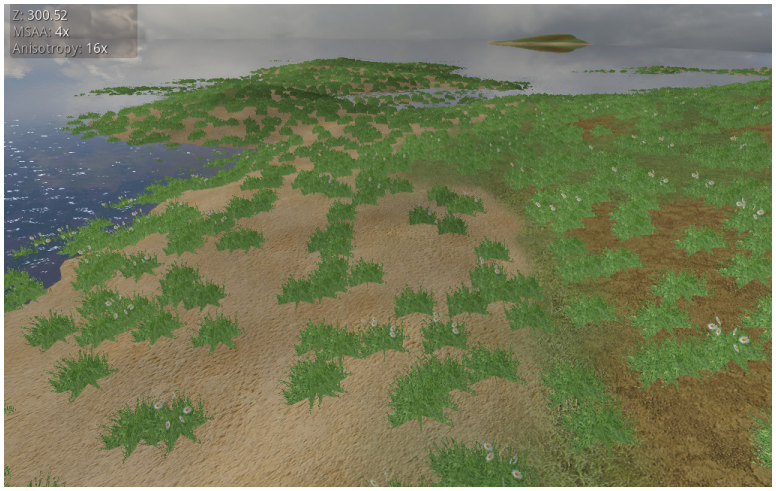
Slika 4.9: Prikaz ručnog uređivanja volumnog terena dodavanjem i oduzimanjem gustoće.

s prozirnošću. Na slici 4.10 može se vidjeti granica između prozirnih i neprozirnih dijelova objekta. Negativna strana ovog pristupa je što ne radi s *antialiasing* metodom višestrukog uzorkovanja, ali budući da je ova tehnika vrlo popularna u računalnim igrama razvijeni su brojni algoritmi koji omogućuju zaglađivanje rubova takvih objekata.

Kod inicijalizacije odabire se gustoća, odnosno maksimalni broj objekata raslinja. Ti objekti se jednolikom razdiobom nasumično raspoređuju po zadanoj površini terena. No takav pristup daje pomalo neprirodne rezultate vidljive na slici 4.11.

Da bi se uklonio taj problem koristi se ranije implementirana funkcija terena koja za bilo koju točku daje boju terena u toj točki. Objektima raslinja definira se raspon

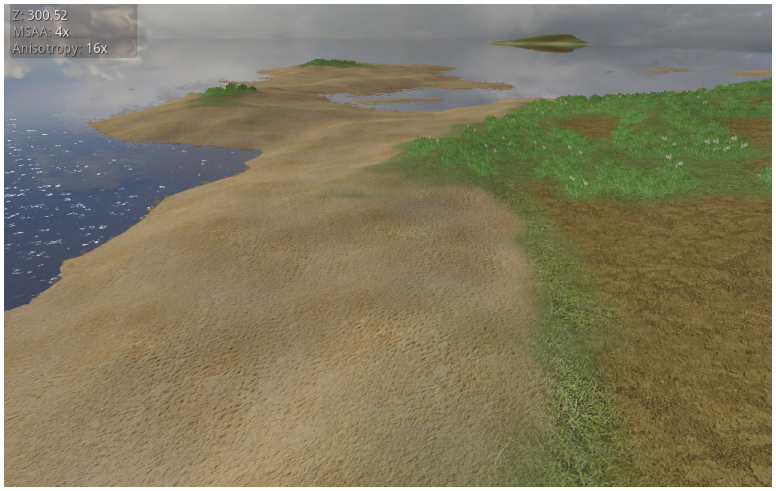
Slika 4.10: Prikaz prozirnosti raslinja.



Slika 4.11: Jednolika razdioba raslinja na terenu.

boja terena na kojima se smije generirati. Tako se definira da visoka trava "raste" samo na travnatim površinama (slika 4.12), a isti način bi se mogli definirati drugi objekti za ukrašavanje drugih vrste terena, primjerice kamenčići na zemljanim površinama.

Kod iscrtavanja scene potrebno je nekad nacrtati i tisuće objekata raslinja. Crtanje svakog objekta zasebno nepotrebno bi trošilo resurse procesora budući da svaki objekt raslinja ima isti model, a razlikuju se jedino prema položaju, rotaciji i teksturi. Zbog toga se koristi instancirano crtanje. Kod inicijalizacije bloka raslinja stvaraju se transformacijske matrice svih objekata i spremaju se redom u spremnik na grafičkom



Slika 4.12: Jednolika razdioba raslinja s ograničenjem prema boji terena.

sklopovlju. Uz matricu sprema se i redni broj teksture koju koristi svaki objekt.

Različite teksture koriste se radi povećanja raznolikosti izgleda raslinja. S obzirom na to da je model koji se crta jednostavan, moguće je samo prosljeđivanjem većeg broja tekstura u jednom prolazu nacrtati mnogo različitog vrsta bilja bez ikakvog pada performansi. Prikaz tekstura trave s cvijećem i bez njega vidi se na slici 4.13.

Slika 4.13: Prikaz visoke trave i cvijeća.

Kod crtanja koristi se naredba OpenGL-a *glDrawElementsInstanced* kojoj se šalje željeni broj objekata za iscrtavanje. Grafički sustav u programu za sjenčanje tada postavlja varijablu koja sadrži redni broj svake instance. Ta varijabla se koristi kao

indeks za dohvaćanje transformacijske matrice i rednog broja teksture iz spremnika.

Da bi prikaz raslinja bio manje statičan gornji dio modela se u programu za sjenčanje vrhova pomiče u zadanom smjeru čime se oponaša utjecaj vjetra. Matematički je pomak definiran kao:

$$polozaj = polozaj + smjer \cdot amplituda \cdot sin(t \cdot brzina + instanceID)$$

Smjer, amplituda i brzina su karakteristike "vjetra" i definirane su kao uniformne varijable, odnosno konstantne su tijekom jedne operacije crtanja, što u slučaju crtanja instanciranih objekata znači da su iste za sve objekte. U svakom objektu razlikuju se položaj i *instanceID*, tj. varijabla koja sadrži redni broj instance. Koristi se kao pomak u funkciji sinus jer bi se inače svi objekti micali paralelno i istovremeno, a ovako se dobiva dojam da se svaki objekt miče slobodno.

4.4. Prikaz vode

Za prikaz vode koristi se jednostavan model u kojem je vodena površina prestavljena ravninom. Kod svakog crtanja okvira čitava se scena crta dva puta. Prvo se crta reflektirana scena u posebnu teksturu koja ima istu rezoluciju kao i prozor koji se prikazuje. Prije tog crtanja transformacijska matrica kamere se skalira tako da faktor skaliranja y osi bude -1. Na taj način dobivamo okrenutu scenu. Kamera se tada pomiče u novi položaj ovisno o visini razine vode tako da prikaže istu scenu, ali iz reflektirane perspektive.

Nakon toga se crta glavna scena. Kod crtanja vode na ravninu vode se perspektivno iz točke kamere projcira tekstura reflektirane scene. Rezultat je prikazan na slici 4.14. Može se vidjeti da postoje greške u prikazu. Objekti ili dijelovi objekata koji se nalaze ispod vode i koji se ne bi trebali reflektirati crtaju se u reflektiranoj slici.

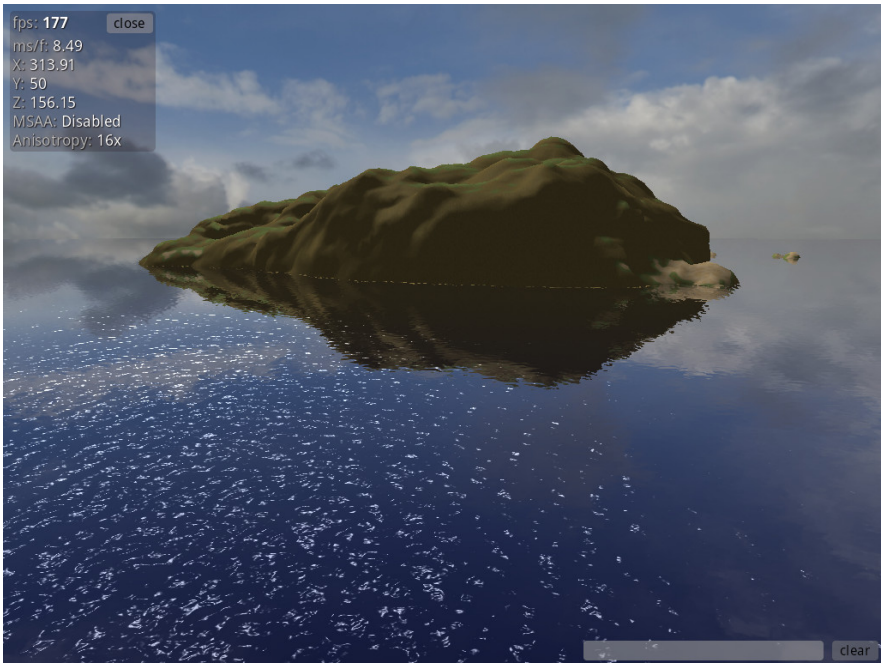
Taj problem se najlakše rješava postavljanjem ravnine rezanja koja odbacuje sve točke koje se nalaze ispod te ravnine. Za ravninu rezanja postavlja se razina vode i na slici 4.15 može se vidjeti prikaz bez artefakata uzrokovanih refleksijom podvodnih objekata.

Ovakav prikaz nema artefakata, ali nije uvjerljiv prikaz vodene površine jer je reflektirana slika ravna, poput zrcala. Zbog toga se na vodu dodaje mapa normale koja izobličava refleksiju. Takva refleksija je prestatična, pa se mapa normale vode uzorkuje više puta s različitim koordinatama tekstura koje su različito skalirane i pomiču se u različitim smjerovima tijekom vremena. Na taj način dobiva se animirana simulacija vodenih valova koji izobličavaju refleksiju. Isto izobličavanje se radi i sa zrcalnom

Slika 4.14: Prikaz artefakata kod crtanja reflektirane scene.

Slika 4.15: Prikaz refleksije.

komponentom reflektirane svjetlosti te se tako postiže uvjerljiva refleksija svjetlosti na vodi. Konačan rezultat vidljiv je na slici 4.16.



Slika 4.16: Prikaz neravne refleksije vode.

4.5. Efekti

Da bi se postigao uvjerljiv ili zanimljiv prikaz scene potrebno je koristiti i dodatne efekte. Na slici 4.17 prikazana je podvodna scena u kojoj se koristi efekt dubine vidnog polja (engl. *depth of field*). Efekt je napravljen kao troprolazni program za sjenčanje u prostoru slike. U prva dva prolaza se slika zamućuje prvo horizontalno pa vertikalno, a u trećem prolazu se slici daje plavičasti ton, koji je na slici ublažen radi bolje preglednosti.

Na slici 4.18 prikazano je preslikavanja sjena na raslinju. Za preslikavanje sjena potrebno je prvo prikaz dubine scene iz perspektive izvora svjetla spremiti u dubinsku teksturu, a kod crtanja provjeravati je li točka dubina točke koju crtamo (u sustavu svjetla) veća od dubine spremljene u dubinskoj teksturi. Ako jest, tada je točka koju crtamo u sjeni i nju ne osvjetljavamo difuznom i zrcalnom komponentom svjetlosti. Za uvjerljiv prikaz sjena potrebno je imati niz dubinskih tekstura koje pokrivaju različita područja i pomoću tih tekstura pokriti čitavo vidno polje.

Za bolji dojam prikaza terena trebalo bi implementirati i druge efekte, poput magle ili različitih vremenskih nepogoda.

Slika 4.17: Prikaz podvodnog zamućenja.

Slika 4.18: Prikaz preslikavanja sjena.

5. Zaključak

U radu su predstavljene metode za prikaz terena zasnovanih na volumnim i visinskim mapama, algoritmi za generiranje slučajnog šuma, volumnih mapa i poligonalnih mreža, kao i metode za teksturiranje obaju vrsta terena. Kod implementacije posebna je pozornost posvećena izvođenju u čim kraćem vremenu i uz što manje korištenje memorije. Stoga su neki algoritmi, poput pokretnih kocki i generiranja teksture boja, implementirani na grafičkom sklopovlju. Implementacija pokretnih kocki na grafičkom sklopovlju omogućila je uređivanje volumnih terena u stvarnom vremenu.

Glavni problem kod izrade volumnih terena bilo je vrijeme potrebno za generiranje volumnih mapa blokova što je riješeno generiranjem bloka funkcije šuma unaprijed, a kod generiranja blokova se uzorkovao već spremljeni šum umjesto da se funkcija šuma ponovno računa za svaku točku svakog bloka terena.

Napravljena je i implementacija visoke trave i cvijeća pomoću instanciranog crtanja da se izbjegne suvišan rad procesora, a također je napravljen i jednostavan sustav animacije tih objekata koji oponaša puhanje vjetra. Dan je pregled implementacije reflektivne vodene površine kao i simulacije vodenih valova.

LITERATURA

OpenGL Architecture Review Board. *OpenGL reference pages, version 4.2*. 2011. URL http://www.opengl.org/sdk/docs/man4/.

Paul Bourke. *Polygonising a scalar field*, 1994. URL http://local.wasp.uwa.edu.au/~pbourke/geometry/polygonise/.

Willem H. de Boer. Fast terrain rendering using geometrical mipmapping. 2000. URL http://www.flipcode.com/archives/article_geomipmaps.pdf.

Hugo Elias. *Perlin Noise*. URL http://freespace.virgin.net/hugo.elias/models/m_perlin.htm.

Alain Fournier, Don Fussell, i Loren Carpenter. Computer rendering of stochastic models. *Commun. ACM*, 25(6):371–384, 1982. URL https://blog.itu.dk/MPGG-E2010/files/2010/09/1011853781.pdf.

William E. Lorensen i Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph.*, 21(4):163–169, 1987. URL http://doi.acm.org/10.1145/37402.37422.

Dimitris Papavasiliou. Color-based per-pixel blending of detail textures. 2011. URL http://www.nongnu.org/techne/research/blending/blending.pdf.

Postupci prikaza terena

Sažetak

U rad je dan pregled metoda generiranja i prikaza terena zasnovanih na visinskim i volumnim mapama. Posebno se obrađuju metode stvaranja poligonalnih mreža i teksturiranja za obje vrste terena koje su izvedive u realnom vremenu. Opisani su postupci instanciranog crtanja detalja na terenu i prikaza reflektivne vodene površine.

Ključne riječi: računalna grafika, teren, visinske mape, volumne mape, pokretne kocke.

Methods for displaying terrain

Abstract

This paper provides a description of methods for generating and displaying height map and volume map based terrains, with a focus on methods for creating polygon meshes and terrain texturing in real time. It also describes methods for instanced rendering of terrain details and rendering of reflective water surfaces.

Keywords: computer graphics, terrain, height maps, volume maps, marching cubes.