SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 2445

Postupci izrade i primjene mapa normala

Andrija Stepić

Zagreb, lipanj 2012. godine

Sadržaj

1. Uvod

Ljudi vole komunicirati i prenositi vlastita mišljenja i doživljaje drugima. Od prvih slika na zidovima špilja, preko razvoja slikarstva, kiparstva, filma i animacije, vizualne reprezentacije događaja su postajale sve realnije i ljudi su mogli doživjeti emocije i vidjeti točno što je netko zamislio, čak i kroz vremenski period, a sve na temelju ljudske mašte. No kako je ljudima uvijek zabavnije razmatrati nešto zahtjevnije i nemoguće umjesto zadovoljiti se postojećim, znanost je napredovala i jedan od vizualno privlačnijih produkata je računalna grafika. Računalna grafika omogućava (između ostalog) prikazivanje nerealističnih svjetova i događaja, te uz puno truda i bez ljudskih glumaca može ispričati priču ili pružiti nebrojeno satova zabave u računalnim igrama.

Kako bi prikaz svjetova, likova, digitalne flore i faune bio što realističniji, potrebna je jača računalna oprema i veći broj detalja pohranjen u podacima koji čine neku grafičku scenu. No budući da nisu svi ljudi u ekonomskoj prilici imati najmoćnija računala, koja i sama imaju granicu kvalitete grafičkog prikaza, iznimna količina detalja u prikazima objekata se postiže na različite načine, koji ne uključuju modeliranje dodatnih detalja na prikazanim objektima, već primjene mape normala i visinske mape.

Tema ovog rada je povezana uz dva srodna koncepta primjene tekstura: visinske mape i mape normala. Cilj rada je proučiti izradu mapa na temelju različitih ulaznih parametara u različitim okruženjima, ograničenja u smislu kvantizacije podataka zapisanih u mape i moguća proširenja, te primjene mapa normala i visinskih mapa za različite znanstvene i druge svrhe. Uz navedeno, prikazuju se usporedbe u kvaliteti prikaza s obzirom na broj poligona u sceni, rezoluciju i način primjene mapa, te usporedbe statičnih scena sa i bez korištenja obje vrste mapa.

Kao praktični dio ovog rada, izrađene su dvije aplikacije, čija svrha je omogućiti dobivanje visinskih mapa iz modela, te mapa normala iz bilo koje teksture. Aplikacije su međusobno nezavisne, te pružaju jednostavno

grafičko sučelje za promjenu parametara, učitavanje sadržaja te spremanje željenog produkta za daljnje korištenje.

2. Model objekta kao skup podataka i tangencijalni prostor

2.1. Model objekta

U računalnoj grafici, modeliranje je proces razvijanja matematički diskretne reprezentacije površine ili objekta. Rezultat modeliranja je model koji se može iz trodimenzinalnog prostora prikazati projekcijom na dvodimenzionalnu ravninu ekrana procesom koji se zove izrada prikaza (engl. rendering). Model se sastoji od više kategorija podataka od kojih su najbitnije prostorne točke koje predstavljaju položaj kvatiziranih točaka modela koje su međusobno povezane u trokute, četverokute, linije, zakrivljene površine i na druge načine. Modeli nastaju „ručnim“ modeliranjem i manipuliranjem točaka u prostoru, algoritamskim određivanjem položaja točaka ili skeniranjem podataka (objekata) iz stvarnog svijeta. Unatoč postojanju više kategorija 3D modela, ovaj rad se fokusira na modele ljuske objekata, koji sadrže podatke same granice (ili vanjske ljuske) objekta, ne pružajući pogled u samu unutrašnjost objekta i dajući privid ispunjenosti prostora. Razlog za ograničenje na granične modele dolazi iz najčešće upotrebe visinskih i mapa normala za poboljšanje vizualnog prikaza na upravo tu kategoriju 3D modela, iako je sama upotreba tih vrsta mapa neograničena na kategoriju modela.

3D modeli se pohranjuju u različitim zapisima na računalu, s osnovnom podjelom na tekstualne - ljudski čitljive i binarne - računalno čitljive formate. Najpoznatiji i najčešći format za pohranu 3D modela je Wavefront .obj format, koji je izvrsno ljudski čitljiv i može sadržavati većinu vrsta podataka o modelu, koji se navode u nastavku.

Spomenuto je da model predstavlja objekt ili površinu dobvenu kvantizacijom i interpretacijom odnosa prostornih točaka. U nastavku ovog poglavlja se daje uvid u općenite podatke modela. Pretpostavimo da su točke povezane u niz trokuta, jer se svaki model može pretvoriti u konačan broj trokuta, najjednostavniju jedinicu dvodimenzionalne povezanosti najmanjeg broja točaka. Svaki trokut modela ima 3 točke s 3 prostorne koordinate.

Slika 2.1 prikazuje trokut u prostoru koji se sastoji od točaka A (0,1,0), B (0,0,1) i C (1,0,0), što bi u zapisu modela u formatu Wavefront .obj rezultiralo zapisom:

...

v 0 1 0

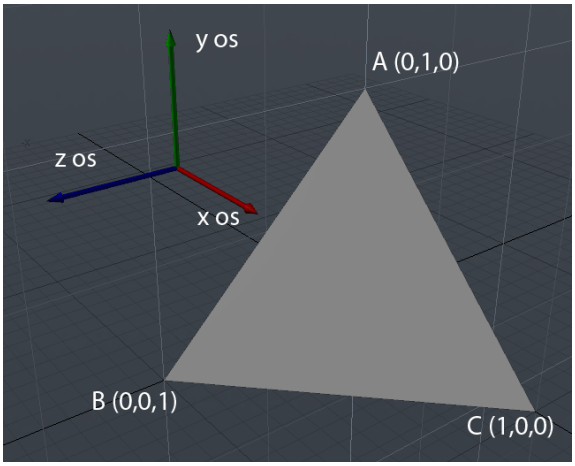
v 0 0 1

v 1 0 0

...

Iz zapisa je vidljivo da se položaj pojedine točke zapisuje u formatu:

v [x koordinata] [y koordinata] [z koordinata]



Slika 2.1.1 – Prikaz trokuta u 3D prostoru s naznačenim osima i koordinatama točaka

Budući da moramo nekako grupirati točke u trokute, za svaki trokut postoje indeksi koji predstavljaju redni broj točke koja pripada tom trokutu.

Slika 2.2 prikazuje primjer indeksiranja točaka trokuta za model tetraedra. U nastavku je dan zapis modela u formatu Wavefront .obj koji sadrži položaje točaka i indekse točaka koji čine lica (engl. face) ili trokute u ovom slučaju. Na slici su korištena imena točaka: t0, t1, t2, t3, gdje broj odgovara indeksu točke u implicitnom nizu točaka u zapisu objekta.

...

v 0 -0.578 -1

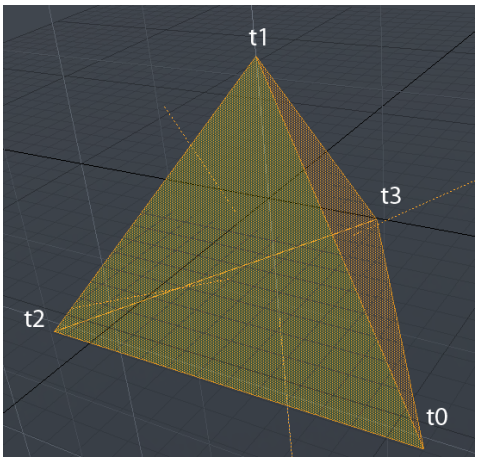
v -0.866025 -0.578 0.5

v 0.866025 -0.578 0.5
v 0 0.578 0
...
f 1 2 4
f 1 3 2
f 3 1 4
f 2 3 4
...

Iz ispisa je vidljivo da indeksiranje točaka počinje od 1, a ne 0, što znači da točka sa indeksa t[i] na slici 2.2 odgovara indeksu [i+1] u zapisu modela. Jedan poligon (engl. face) se zapisuje u formatu:

f [indeks 1. točke] [indeks 2. točke] [indeks 3. točke] ...

ukoliko su dostupni samo podaci o položajima točaka.



Slika 2.1.2 – Prikaz indeksa modela tetraedra

Svaka točka može imati vlastite UV koordinate, 2D koordinate koje predstavljaju jedinstveno preslikavanje točke modela u dvodimenzionalnu ravninu, a služe povezivanju točaka modela s koordinatama slikovnog elementa tekstura različitih primjena. Za precizno određivanje položaja

slikovnog elementa teksture na određenoj točci poligona, koristi se interpolacija. Osnovni prostor za UV koordinate modela je identičan prvom kvadrantu 2D kartezijevog koordinatnog sustava, ograničenog na kvadrat veličine stranice 1. Naravno, točke mogu imati proizvoljne UV koordinate, što u primjeni rezultira slaganjem teksture u mozaik i određivanjem u koju koordinatu osnovnog UV kvadranta (između točaka (0,0) i (1,1)) se preslikava neka točka koordinata modela.

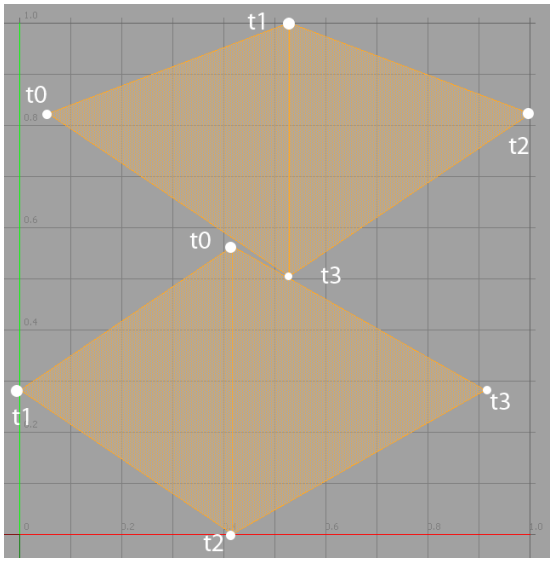
Slika 2.3 prikazuje prethodni model tetraedra preslikanog na vlastitoj UV mapi. Prema slici, točke tetraedra su preslikane u spomenuti osnovni kvadrant UV prostora. Na slici su dodatno naznačeni indeksi točaka koji odgovaraju prikazu modela sa slike 2.2, te je iz njih vidljivo da točke koje su bile zajedničke pojedinim poligonima modela imaju više različitih preslikanih UV koordinata.

U nastavku je prikazan ispis UV koordinata danog modela u formatu Wavefront .obj.

```
...
vt 0.91165 0.281566
vt 0.416098 0.563131
vt 0.416098 0
vt 0.527402 0.504009
vt 1 0.822333
vt 0.527402 0.999817
vt 0 0.281566
vt 0.0548048 0.822333
...
```

Iz ispisa je vidljivo da je zapis pojedine UV koordinate oblika:

vt [x koordinata točke] [y koordinata točke]



Slika 2.1.3 – Prikaz modela tetraedra preslikanog na 2D UV koordinate

Budući da je trokut najjednostavniji poligon, trokut ima vektor normale na ravninu koja ga sadrži. Normale modela mogu biti pohranjene za svaki trokut, ili za svaku točku. Budući da upotreba normala ovisi o modelu osvjetljivanja i sjenčanja scene, navedimo sve mogućnosti. Matematički, svaka ravnina ima dva jedinična vektora normale, koji su međusobno suprotnog smjera. Kako su tri točke trokuta dovoljne da se odredi ravnina u kojoj leži trokut, tako se iz istih točaka mogu odrediti i vektori normale.

Jednadžba ravnine je $Ax + By + Cz + D = 0$

Gdje se brojevi (A,B,C) mogu protumačiti kao vektor normale na ravninu (ne nužno jedinični), a sve skupine koordinata (x,y,z) koje zadovoljavaju jednadžbu su točke koje pripadaju ravnini.

Općenito, točke trokuta se mogu zadati u redoslijedu kazaljke na satu i redoslijedu suprotnom od kazaljke na satu, gdje spomenuti redoslijed utječe na izbor između dva moguća jedinična vektora normale na trokut (i njegovu ravninu) ukoliko se normala računa sa istim redoslijedom točaka u formuli. U slučaju 3D točaka (t1,t2,t3) koje su zadane u redoslijedu kazaljke na satu, normala na dani trokut se određuje na slijedeći način:

Normala = (t2 – t1) x (t3 – t1)

Gdje je „x“ vektorski produkt, a vektor normale po komponentama glasi

Normala.x = (t2.y-t1.y)(t3.z-t1.z) – (t3.y-t1.y)(t2.z-t1.z)

Normala.y = (t2.z-t1.z)(t3.x-t1.x) – (t3.z-t1.z)(t2.x-t1.x)

Normala.z = (t2.x-t1.x)(t3.y-t1.y) – (t3.x-t1.x)(t2.y-t1.y)

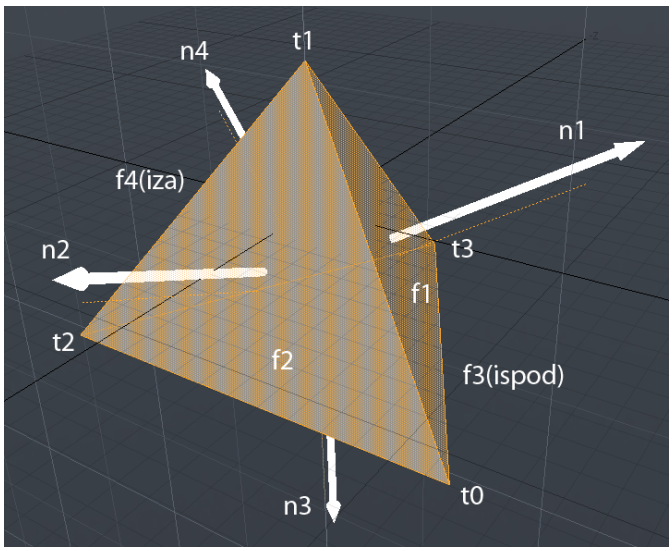
Spomenuto je da točka može imati više normala, što je preuzeto iz situacije da točka pripada ne jednom, nego više trokuta, te time preuzima njihove normale. Da bi se dobila ispravna normala u točci koju dijeli više trokuta, zasebne normale tih poligona se mogu zbrojiti u aritmetičku sredinu, te će se dobiti vektor normale za točku koji je interpolacija normala poligona. Daljnja upotreba vektora normala točke opisana je u podpoglavlju „Primjena mapa normala“.

Zbog jednostavnosti, slika 2.4 prikazuje naznačene točke, poligone i normale poligona koji čine model tetraedra, te je u nastavku dan ispis normala istog modela u formatu Wavefront .obj.

...
vn 0 -1 0
vn -0.79486 0.396983 -0.458913
vn 0 0.396983 0.917826
vn 0.79486 0.396983 -0.458913
...

Iz ispisa je vidljivo kako je format zapisa jedne normale oblika:

vn [x koordinata] [y koordinata] [z koordinata]



Slika 2.1.4 – prikaz modela tetraedra s naglaskom na normale poligona

Ukoliko model sadrži krivulje ili zakrivljene površine, može sadržavati podatke o prostornim točkama krivulje i težinskim faktorima točaka, no ovdje se ograničavamo na jednostavne modele sastavljene isključivo od trokuta.

Dodatno, poligoni modela mogu biti podjeljeni u grupe sa posebnim namjenama, kao što je hijerarhijska struktura u svrhu animacije, ili grupe koje definiraju poligone za transformaciju u zaglađene površine i slično.

Svaki trokut modela može imati primjenjen određeni materijal, koji sadrži niz fizikalnih svojstava i svojstava scene potrebnih za izračunavanja utjecaja objekata osvjetljenja ili samog modela na konačni intenzitet svake točke tog trokuta. Najpoznatija svojstva su ambijentalna boja, difuzna boja, reflektivna boja odsjaja, reflektivni koeficijent, transparentnost, indeks refleksije, boja refleksije, boja transparentnosti, koeficijent disperzije, boja i koeficijent iluminacije i drugi.

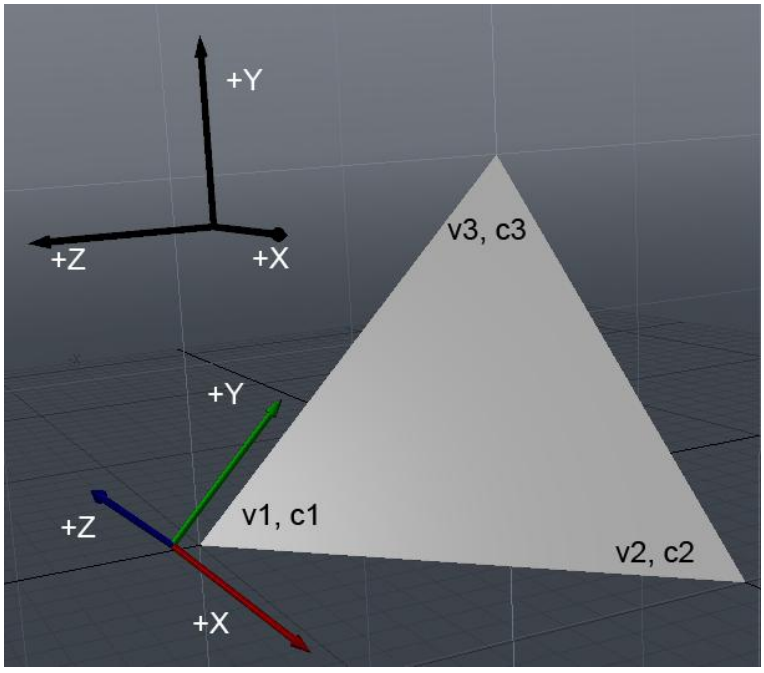
2.2. Tangencijalni prostor

U prethodnom dijelu je objašnjeno kako se model sastoji od točaka koje su sastavni dijelovi poligona, te se koordinate točaka pohranjuju s obzirom na njihov položaj u prostoru scene. Postoje tehnike primjene tekstura na model koje zahtijevaju podatke o tangencijalnom prostoru svakog poligona, kako bi se vektori drugih objekata (svjetla, kamera) mogli interpolirati između točaka s obzirom na njihov tangencijalni prostor.

Tangencijalni prostor se može zamisliti kao koordinatni sustav koji pripada nekoj lokalnoj površini. Glavni vektori osi tog prostora su normala, koja je okomita na tu površinu, te tangenta i bitangenta koje su okomite na normalu. Iznosi vektora osi ovise o geometriji točaka površine modela u 3D prostoru scene, te koordinatama UV prostora istih točaka. Smjer vektora tangente 3D prostora jednak je smjeru rastuće U koordinate UV prostora (ekvivalentno X osi), a smjer bitangente jednak je smjeru porasta V

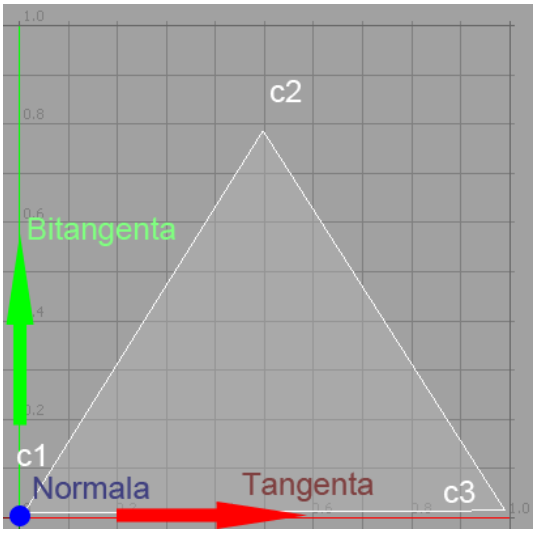
koordinate (ekvivalentno Y osi). Normala je ekvivalent Z osi koja uvijek ide u pozitivnom smjeru iz površine. Spajanjem navedena tri vektora nakon normaliziranja dobiva se ortonormalna matrica rotacije čija svrha je pretvaranje vektora tangencijalnog prostora u prostor scene ili obrnuto. Budući da matrica sadrži normalizirane vektore koji predstavljaju osi lokalnog (tangencijalnog) prostora neke površine, ona zapravo ima ulogu rotacije vektora u novi prostor, koji se može podudarati s prostorom scene.

Najjednostavniji način dobivanja tangencijalne matrice (u nastavku TBN matrice) je izračunom iz podataka modela trokuta koji se, naravno, sastoji od 3 točke, normale, te 3 koordinate UV prostora. Slika 2.2.1 prikazuje koordinatne osi prostora scene crnim strelicama, te osi tangencijalnog prostora trokuta strelicama u boji.



Slika 2.2.1 – Prikaz osi tangencijalnog prostora i prostora scene

Slika 2.2.2 prikazuje UV mapu trokuta prikazanog u prostoru scene na slici 2.2.1. Vektori tangencijalnog prostora su označeni na osima; Z os ide iz ekrana, okomito na druge dvije osi.



Slika 2.2.2 – Prikaz UV prostora i njegovih osi

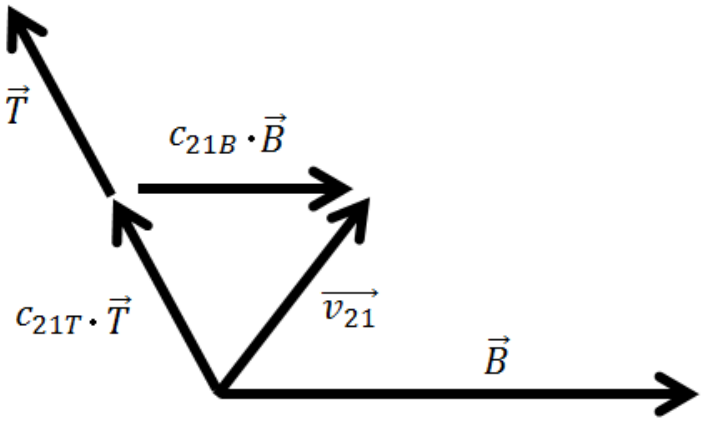
Neka su podaci trokuta nazvani kao na slici 2.2.1, slijedećim redoslijedom: koordinate u prostoru scene: v_1, v_2, v_3, njihove odgovarajuće dvodimenzionalne koordinate UV prostora: c_1, c_2, c_3, te neka je normala poligona poznata, i imenovana vektorom $\vec{N}$. U daljnjem opisu pretpostavlja se da je izgled UV mape prikazanog trokuta jednak kao prikaz trokuta na slici 2.2.1. Sa slike se može primijetiti da vektor od točke v_1 do v_2 ima isti smjer u prostoru scene kao i vektor od točke c_1 do c_2 u UV prostoru. Isto vrijedi i za točke scene v_1 i v_3, te točke UV prostora c_1 i c_3. Neka vrijedi slijedeće:

$$\overrightarrow{v_{21}} = v_2 - v_1, \quad \overrightarrow{v_{31}} = v_3 - v_1, \quad \overrightarrow{c_{21}} = c_2 - c_1, \quad \overrightarrow{c_{31}} = c_3 - c_1$$

Komponente vektora nastalih razlikom točaka scene su x, y i z, a komponente vektora nastalih razlikom točaka UV prostora označavaju se na slijedeći način:

$$\overrightarrow{c_{21}} = (c_{21T}, c_{21B})$$

Komponente točke UV prostora se u pravilu označavaju s (x,y) ili (u,v), no u ovom primjeru će se označavati s (T,B) jer vektor $\vec{T}$ predstavlja x ili u os, a vektor $\vec{B}$ predstavlja y ili v os. Potrebno je odrediti vektore tangente (u nastavku $\vec{T}$) i bitangente (u nastavku $\vec{B}$). Slika 2.2.3 prikazuje vezu između vektora točaka UV prostora, vektora $\vec{T}$ i $\vec{B}$ te vektora između točaka prostora scene.



Slika 2.2.3 – Prikaz odnosa podataka trokuta i vektora tangencijalnog prostora

Iz prikazanog odnosa mogu se formulirati slijedeće dvije jednadžbe:

$$\overrightarrow{v_{21}} = c_{21T} * \vec{T} + c_{21B} * \vec{B}, \overrightarrow{v_{31}} = c_{31T} * \vec{T} + c_{31B} * \vec{B}$$

Ovaj sustav dvije jednadžbe s dvije nepoznanice može se zapisati kao:

$$\begin{bmatrix} \overrightarrow{v_{21}} \\ \overrightarrow{v_{31}} \end{bmatrix} = \begin{bmatrix} c_{21T} & c_{21B} \\ c_{31T} & c_{31B} \end{bmatrix} \begin{bmatrix} \vec{T} \\ \vec{B} \end{bmatrix}$$

Rješenje sustava su dva linearno nezavisna vektora baze tangencijalnog prostora:

$$\vec{T} = \frac{1}{c_{21T} * c_{31B} - c_{31T} * c_{21B}} * (c_{31B} * \overrightarrow{v_{21}} - c_{21B} * \overrightarrow{v_{31}})$$

$$\vec{B} = \frac{1}{c_{21T} * c_{31B} - c_{31T} * c_{21B}} * (c_{21T} * \overrightarrow{v_{31}} - c_{31T} * \overrightarrow{v_{21}})$$

Budući da je vektor normale $\vec{N}$ okomit na tangentu i binormalu, dobiva se jednostavnim vektorskim produktom:

$$\vec{N} = \vec{T} \, x \, \vec{B}$$

Komponente triju određenih vektora imaju značenje jediničnih vektora osi tangencijalnog prostora te čine tangencijanu matricu TBN na slijedeći način:

$$Matrica_{TBN} = [\vec{T} \quad \vec{B} \quad \vec{N}] = \begin{bmatrix} T_x & B_x & N_x \\ T_y & B_y & N_y \\ T_z & B_z & N_z \end{bmatrix}$$

Dobivena matrica množenjem vektora tangencijalnog prostora s lijeve strane daje vektor prostora scene, no u primjeni u programima za sjenčanje potrebno je imati matricu koja radi upravo obrnuto: pretvara vektor (ili točku)

prostora scene u tangencijalan prostor. To se postiže inverznom TBN matricom. Budući da vektori baze nekog prostora trebaju biti jedinični, dobivene vektore potrebno je normalizirati. Vektori baze prostora trebaju biti međusobno okomiti, no u praksi vektori $\vec{T}$ i $\vec{B}$ često nisu okomiti, te se Gram-Schmidtovim algoritmom vrši korekcija njihovih okomitosti, čuvajući okomitost s vektorom normale. Korekcija glasi:

$$\overrightarrow{T'} = \vec{T} - (\vec{N}\ dot\ \vec{T}) * \vec{N},\ \overrightarrow{B'} = \vec{B} - (\vec{N}\ dot\ \vec{B}) * \vec{N} - (\overrightarrow{T'}\ dot\ \vec{B}) * \overrightarrow{T'}$$

Gdje je operator '*dot*' skalarni umnožak, a operator '*' umnožak skalara s vektorom. Nakon ove korekcije, vektori na prije navedeni način tvore matricu TBN. Zbog ortogonalnosti jediničnih vektora, matrica TBN je ortonormalna, te joj je inverz jednak transponiranoj matrici, pa se pretvorba vektora scene ($\overrightarrow{v_s}$) u vektor tangencijalnog prostora ($\overrightarrow{v_t}$) računa na slijedeći način:

$$\overrightarrow{v_t} = \overrightarrow{v_s} * M_{TBN}^{-1} = \overrightarrow{v_s} * M_{TBN}^{T} = \overrightarrow{v_s} * \begin{bmatrix} \vec{T} \\ \vec{B} \\ \vec{N} \end{bmatrix} = \overrightarrow{v_s} * \begin{bmatrix} T_x & T_y & T_z \\ B_x & B_y & B_z \\ N_x & N_y & N_z \end{bmatrix}$$

Postupak je izveden za model trokuta jer je trokut najjednostavniji poligon čije točke definiraju ravninu. Naravno, tri kolinearne točke ne definiraju ravninu, no model s takvim „trokutom“ je iznimno loš model i ne pojavljuje se u praksi. U slučaju ispravnog trokuta, potrebno je izračunati vektore normale, tangente i bitangente samo jednom za taj trokut, što značajno smanjuje broj izračuna u usporedbi sa slučajem gdje se matrica tangencijalnog prostora mora računati za svaku točku zasebno, jer su poligoni sastavljeni od više točaka koje redovito nisu u istoj ravnini.

Dobivena transponirana TBN matrica se najčešće koristi za transformaciju vektora smjera svjetlosti u programu za sjenčanje točaka, kako bi se dobiveni vektor u tangencijalnom prostoru interpolirao za svaki slikovni element, tako da se u programu za sjenčanje točaka može izračunati ispravni intenzitet osvjetljenja.

3. Visinske mape

U računalnoj grafici, visinska mapa je dvodimenzionalna tekstura u kojoj je pomoću raspona boja pohranjen podatak o visini (elevaciji) površine. Iza prethodne jednostavne definicije skriva se velik broj načina prikaza, izrade i primjena visinskih mapa, od kojih će svaki biti objašnjen uz praktične primjere. U nastavku se uvodno navode tri glavne kategorije primjena visinskih mapa, zbog navođenja u primjerima koji prethode detaljnijem objašnjenju pojedine tehnike. Osnovne primjene visinskih mapa se dijele na:

1. preslikavanje neravnina (engl. bump mapping) u kojem se tekstura koristi za stvaranje detaljnijih sjena na površini,
2. preslikavanje premještanjem geometrije (engl. displacement mapping) u kojem se stvarnoj geometriji modela mijenja položaj na temelju postojećih normala i boje sa teksture
3. generiranje 3D modela na temelju rezolucije i boja tekture

3.1 Način prikaza visinske mape

Stvarni svijet je ispunjen kontinuiranim vrijednostima, a digitalni svijet koji pokušava oponašati stvarni mora se zadovoljiti diskretnom kvantizacijom, pri kojoj se gube detalji između kvantnih razina. Kod visinskih mapa kvantizacija je prisutna u jedinim informacijama koje tekstura može pružati: rezoluciji i boji.

3.2 Boje visinskih mapa

Većina visinskih mapa sastoji se od jednog kanala boje koji se interpretira kao „visina“ ili „udaljenost“ od „dna“ površine. Budući da jedan kanal teksture može sadržavati samo različite iznose sivih boja između crne i bijele (engl. grayscale range), crna boja predstavlja minimalnu, siva neutralnu ili srednju, a bijela maksimalnu visinu. Prilikom primjene visinske mape, obično postoji parametar koji određuje koliku visinsku udaljenost predstavlja 1 stupanj razlike u boji. Visinske mape ne moraju nužno biti spremljene u kanalu teksture, već mogu biti pohranjene u specijaliziranom formatu sa

primjenjenim kompresijama, uz dodatne podatke o načinu primjene, poput spomenute udaljenosti stupnja razlike boje.

S obzirom na dostupne kanale teksture, moguće je iskoristiti i više kanala za pohranu boja koje se interpretiraju kao visinske koordinate površine. Na primjer, standardna 8-bitna tekstura može sadržavati samo 256 različitih vrijednosti sive boje, dakle 256 visinskih vrijednosti. Korištenjem boja, dakle 3 8-bitna kanala boja, broj visina koje se mogu zapisati raste na 256^3 = 16,777,216. Ukoliko se iskoristi i 4., alfa kanal (koji se inače koristi za mjeru transparentnosti pojedinog piksela), broj visina raste na 256^4 = 4,294,967,296. Ovisno o željenoj preciznosti, koristi se neka od navedenih tehnika, odnosno načina zapisa visina, s tim da veća razina detalja dovodi do tekstura koje zauzimaju više memorijskog prostora, čiji iznos varira ovisno o formatu teksture (s gubitkom detalja ili očuvanjem podataka), algoritmu kompresije i naravno samim podacima u teksturi.

U praksi visinske mape pohranjene u jedan kanal boje (256 razina) su iznimno česte zbog jednostavnog razloga – jednostavnost primjene uz najmanje zauzeće memorije. Što se tiče malenog broja razina, vizualni efekt se popravlja većom količinom druge vrste informacije – rezolucijom.

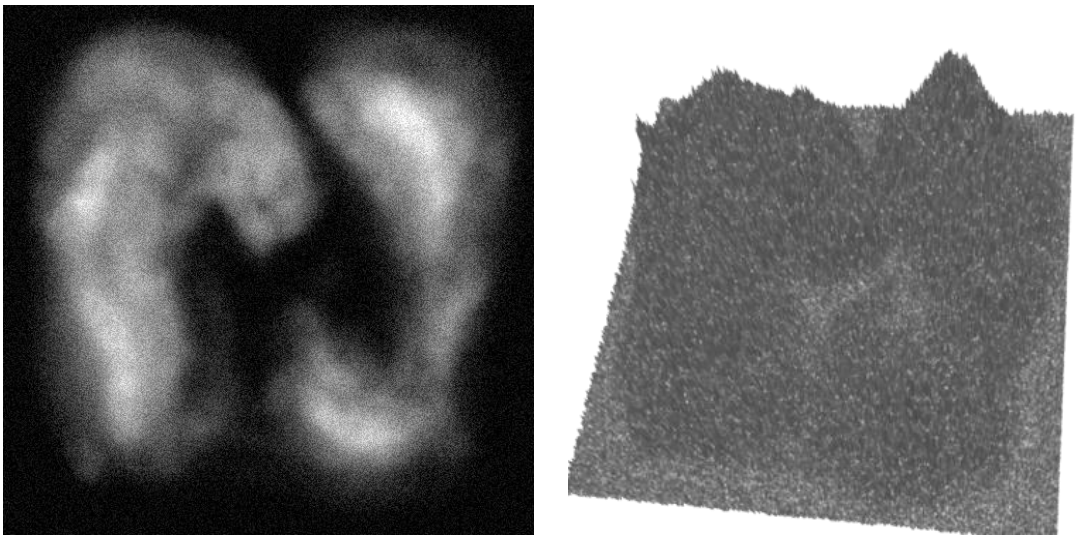
3.3 Rezolucija

Količina razlučivosti detalja mjeri se rezolucijom slike, koja se izražava u količini slikovnih elemenata u širini i visini slike. Zbog povijesnih ograničenja računalnih komponenti, i postupka reduciranja udaljenih tekstura, one su morale biti dimenzija koje su potencije broja 2, no danas sve više prevladavaju komponente koje nemaju to ograničenje, iako se zbog kvalitete prikaza smanjenih tekstura, navike, ustaljenih metoda kompresije i unazadne kompatibilnosti teksture za računalne igre i druge simulacije i dalje stvaraju u dimenzijama pod istim uvjetom.

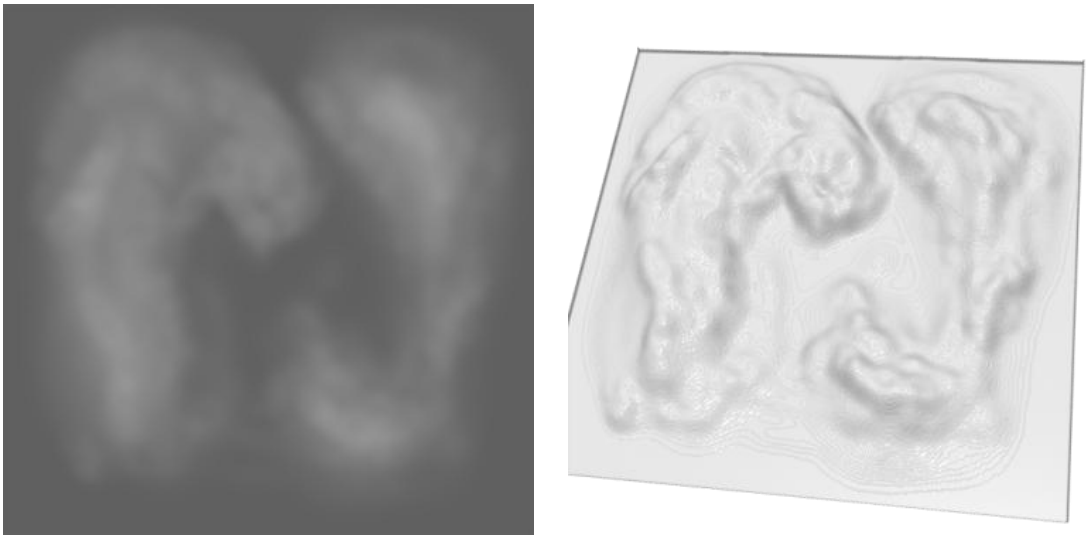
Unatoč logičkoj nepovezanosti između rezolucije i kvantizacije razina visinske mape, za optimalne rezultate potrebno je naći dobar omjer danih vrijednosti, koji će dati zadovoljavajući izgled i performanse. U nastavku su dana dva loša ekstrema u odabiru vrijednosti.

Ako se primjenjuje visinska mapa visoke frekvencije alterniranja intenziteta boja (količina elevacije), te niske rezolucije, prikaz rezultata će izgledati neprirodno, predinamično i neprecizno zbog naglih razlika u boji, a time i simuliranom pomaku geometrije ili intenzitetu sjena. Lijevi dio slike 3.1 prikazuje teksturu navedenih (ne)kvaliteta i njenu primjenu na preslikavanje premještanjem geometrije (desni dio). Primjenjena tekstura je ista tekstura koja će biti prikazana na slici 3.3, uz dodatak uniformnog nasumičnog šuma koji je modulirao intenzitete elevacije za postizanje visoke frekvencije alterniranja u ovom primjeru.

Kao protuprimjer, primjena teksture velike rezolucije a malenih razmaka u vrijednostima visina susjednih slikovnih elemenata će zahtjevati više memorije i dulje procesiranje, a mogla se iskoristiti manja tekstura za približno jednak vizualni učinak. Slika 3.2 sadrži reprezentativne teksture visinske mape (lijevo) i primjene preslikavanjem premještanja geometrije (desno) za navedene vrijednosti.



Slika 3.1 – Primjer visoke frekvencije i niske rezolucije visinske mape



Slika 3.2 – Primjer niske frekvencije i visoke rezolucije visinske mape

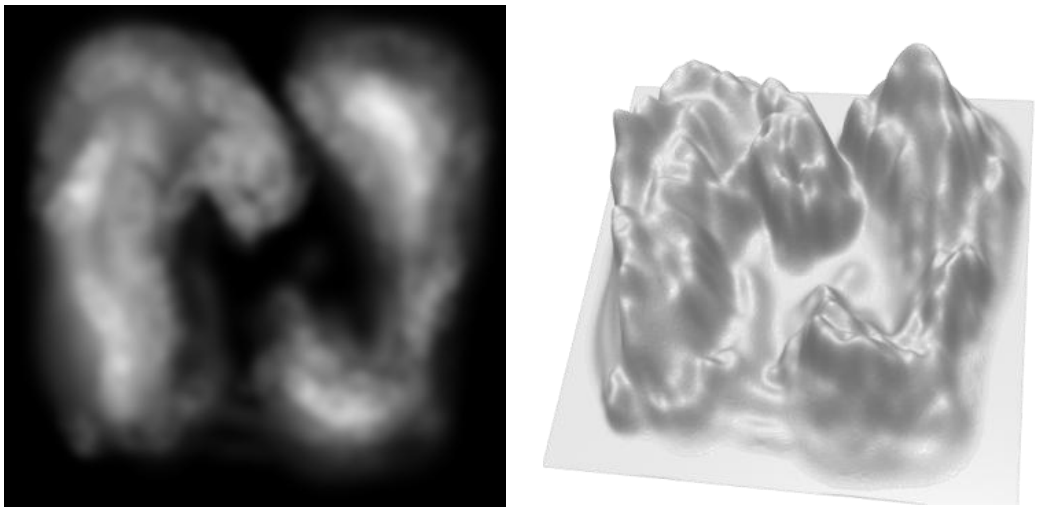
3.4 Stvaranje visinskih mapa

Postoje različiti načini izrade visinskih mapa, koji u kvaliteti variraju od amaterskog isprobavanja u nepoznatom do profesionalnog manipuliranja postojećim podacima. U nastavku ovog poglavlja se navode glavne kategorije izrade visinskih mapa stupnjevito poredane od nepreciznih i nestručnih načina. Potrebno je napomenuti da su navedene one metode koje su dostupne prosječnom korisniku računala s pristupom internetu.

3.4.1. Izrada u programu za manipuliranje slikom

Kako je već spomenuto, visinske mape su uglavnom crno – bijele teksture, koje je moguće uz malo vještine i mašte ručno stvoriti u bilo kojem programu za manipuliranje slikovnim elementima. Na primjer, u programu Photoshop CS5 moguće je stvoriti novu teksturu, odabrati alat nekog oblika, gustoće i toka boje, te naizmjenično nanositi boje u rasponu između crne i bijele (u modelu boja RGBA to bi bile boje koje imaju jednake iznose komponenti R, G i B, odnosno iznose crvene, zelene i plave boje) sve dok se ne postigne željeni učinak. Prednost ovog načina je jednostavnost izrade i dostupnost alata, a nedostatak je nepreciznost, neizraženost i manjak detalja

rezultantne visinske mape. Lijevi dio slike 3.3 prikazuje jednu visinsku mapu dimenzija 512x512 slikovnih elemenata izrađenu na prethodno opisani način u manje od 5 minuta. Svi nedostaci su vidljivi iz slike. Desni dio slike 3.3 prikazuje primjenu teksture s lijevog djela istog prikaza preslikavanjem premještanja geometrije (engl. displacement mapping) na dva trokuta u ravnini XZ u programu Luxology Modo 601.



Slika 3.3 - Visinska mapa i primjena preslikavanjem premještanja geometrije

3.4.2. Algoritamska izrada

Visinske mape terena se često dobivaju primjenom algoritma koji od pseudo-nasumičnih brojeva ili konkretnom matematičkom formulom dobiva niz brojeva, koji se skaliraju i zapisuju u nekom od slikovnih formata, tvoreći manje-više prirodnu reprezentaciju elevacije površine.

U općem slučaju, proceduralno generirane teksture se najčešće dobivaju kombiniranjem više frekvencija vrijednosti interpolacijske ili druge parametarske funkcije kojom se postižu varijacije vrijednosti, tako da konačna tekstura sadrži stupnjevitu razinu detalja koji su u pozadini postignuti matematički, te vizualni rezultat izgleda prirodnije.

3.4.2.1 Perlinov šum

Jedan od popularnijih algoritama za simuliranje prirodnih pojava u računalnoj grafici je Perlinov šum (engl. Perlin noise). Implementacija Perlinovog šuma temelji se na proceduralnom izračunavanju interpolacijske funkcije za sve slikovne elemente teksture, konkretnije njihove položaje.

U nastavku je dan opis općenitog algoritma Perlinovog šuma za proizvoljan broj dimenzija.

1. Definiranje n-dimenzionalne funkcije koja će vraćati pseudo-nasumičan normaliziran vektor za neku točku n-dimenzionalnog prostora: g(x1,x2,...,xn) = (x1a,x2a,...,xna)

2. Definiranje interpolacijske funkcije koja za ulaznu točku iz intervala [0,1] vraća težinski faktor između početne i krajnje točke intervala a izlaz je realni broj. U implementacijama se najčešće koristi linearna funkcija.

3. Za svaku točku se odrede normalizirani vektori od rubnih, početnih točaka do trenutne točke, te se pomnože skalarnim produktom sa vektorima koji su izlaz funkcije iz 1. koraka za ulazne koordinate rubnih točaka, rezultati skalarnih produkta se zbroje

4. Realni broj koji je rezultat koraka 3. interpoliraj između trenutne i okolnih točaka teksture

5. Podjeli prostor i izaberi nove rubne točke, te ponavljaj korake 3-5 do zadovoljavajuće razine.

Prema opisu algoritma je vidljivo da rezultat ovisi o nepredefiniranim funkcijama i načinu podjele prostora (koji se niti ne mora implementirati),

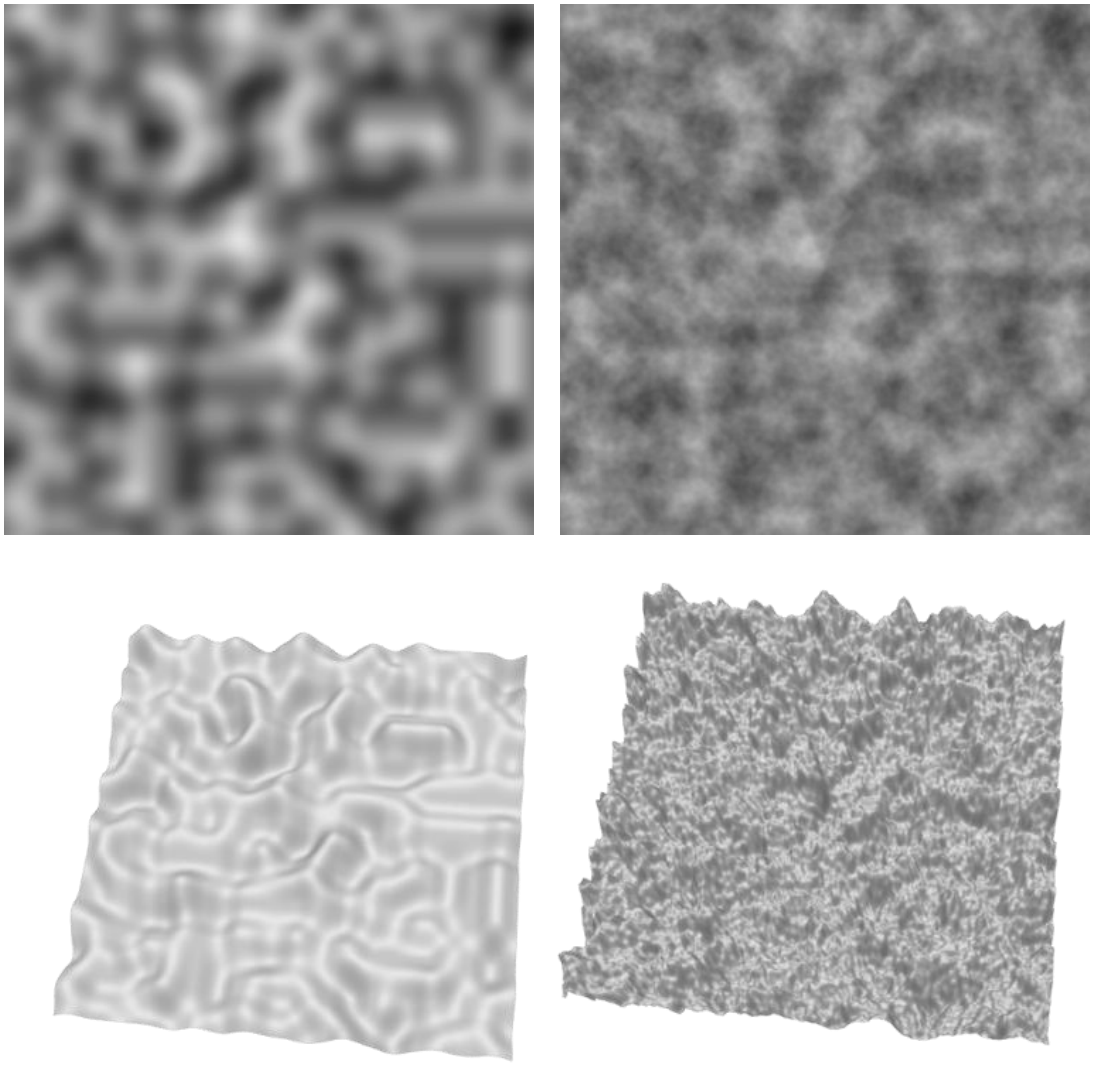
zbog čega postoji velik broj algoritama i rezultantnih tekstura koje svoje temelje nalaze u Perlinovom šumu.
Unatoč naizgled nasumičnom rezultatu, rezultatna tekstura nakon primjene Perlinovog šuma ima specifičnu karakteristiku: svi intervali „svjetlih“ i „tamnih“ područja slike jednake su veličine, poput regularnih „nasumičnih“ intervala pojava u prirodi.

Implementacija Perlinovog šuma ostvarena je na način da se nove teksture mogu dobiti brzo i lagano, uz izmjenu par parametara, a rezultati će i dalje biti međusobnio različiti no prirodni, što ovaj način dobivanja visinskih mapa čini savršenim za slučajeve u kojima je potrebna brza nasumičnost prirodnih simulacija u realnom vremenu uz malo memorijskih resursa.

Slika 3.4 u gornjem redu prikazuje teksture generirane Perlinovim šumom, lijevo gore je tekstura s 1 frekvencijom, desno gore tekstura s 4 frekvencije, a u donjem redu je primjena istih na preslikavanje premještanjem geometrije.

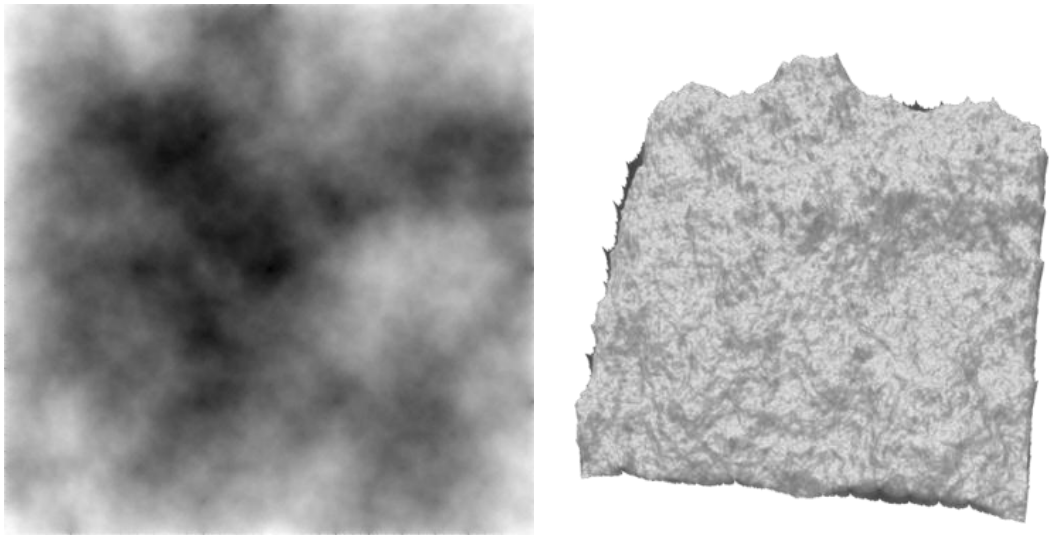
3.4.2.2 Algoritam dijamant - kvadrat

Drugi algoritam koji se često spominje u izradi visinskih mapa je algoritam dijamant-kvadrat (engl. diamond – square algorithm), poznat i pod nazivima fraktal nasumičnog pomaka središnje točke (engl. random midpoint displacement fractal), fraktal oblaka (engl. cloud fractal), fraktal plazme (engl. plasma fractal). Algoritam je jednostavan za implementaciju, no zahtjeva prethodno definirano polje brojeva, čija vrijednost bitno utječe na izgled konačne slike. Postoje razne varijacije ovog algoritma u pokušaju uklanjanja prepoznatljivog oblika koji se pojavljuje zato što svaki korak algoritma stvara ili mijenja iznos slikovnog elementa koji je točno na sredini od prethodnika.
Algoritam se zove dijamant – kvadrat jer izvorne koordinate točaka za svaki korak alterniraju između istoimenih oblika.



Slika 3.4 – Visinske mape nastale Perlinovim šumom i primjena istih

Slika 3.5 prikazuje visinsku mapu nastalu algoritmom dijamant – kvadrat (lijevo) te rezultat njene primjene na preslikavanje premještanjem geometrije.



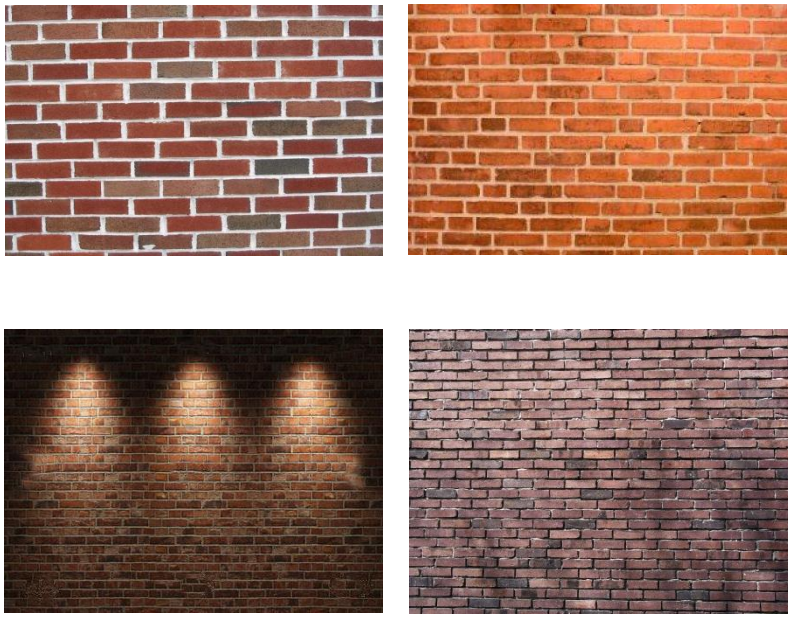
Slika 3.5 - Visinska mapa nastala algoritmom dijamant - kvadrat i primjena te visinske mape na premještanje geometrije modela

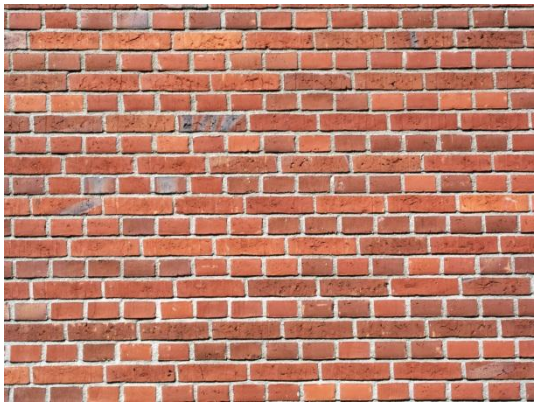
3.4.3. Izrada iz postojećih tekstura

Najčešći slučaj izrade visinskih mapa u industriji računalnih igara je izrada iz tekstura koje predstavljaju stvarni svijet slikan digitalnom kamerom ili 3D čitačima. Izvorne slike su velikom većinom u boji, točnije; zauzimaju 3 kanala = 24 bita informacije po slikovnom elementu. Unatoč tome što je moguće primjeniti visinsku mapu koja je istog formata po slikovnom elementu, boje gotovo nikad neće biti u skladu s visinskom udaljenošću. Ako se razmotri prethodno navedena prednost visinskih mapa sa 8 bita po slikovnom elementu (256 razina boja), jasno je da se javlja potreba za algoritmom konverzije teksture u boji (16777216 boja) u crno bijelu teksturu (256 boja).

3.4.3.1 Izbor teksture

Pri obradi postojeće teksture u visinsku mapu, potrebno je izabrati ispravne teksture. U terminologiji računalne grafike, ispravna se ne odnosi na teksturu koja nema logičke ili podatkovne greške (no takva ispravnost jest preduvjet), već se gleda vizualna ispravnost teksture. Izvorna slika bi trebala biti u što manjoj mjeri perspektivno izobličena, te bi linije na njoj trebale biti što ravnije, i paralelne sa slikom. Osvjetljenje treba biti ravnomjerno, a ne djelomično, raznoliko, ili grupirano na određenim mjestima, te je poželjno da svjetlost pada pod kutom što bližim okomici na površinu. Slika 3.6 prikazuje razne teksture ciglenog zida koje se razmatraju za izvornu teksturu u procesu izrade visinske mape. Neke su perspektivno izobličene (gornji red), jedna ima neispravno osvjetljenje (srednji red, lijevo), kao i slijedeća, koja dodatno sadrži neparalelne linije u odnosu na rub slike (donji red), a posljednja slika je ispravna tekstura za izradu visinske mape (srednji red, desno).





Slika 3.6 – Primjeri 4 neispravne i jedne ispravne teksture ciglenog zida za izradu visinske mape, preuzete sa [8] – [12]

Iz priloženih tekstura se vidi da ni zadnja tekstura nije potpuno ispravna, no savršena tekstura ne postoji, zbog zakrivljenosti leća fotoaparata. Računalna grafika ionako nije savršena, već ***dovoljno dobra*** kvantizacija.

3.4.3.2 Filtar sivih razina

Najjednostavniji način pretvorbe je obrada teksture u boji sa filtrom koji zbraja komponente boja ulaznog slikovnog elementa, dijeli zbroj sa 3 – broj komponenti, te vraća slikovni element koji ima sve 3 komponente boje jednake rezultatu djeljenja – srednjoj vrijednosti komponenti boja ulaznog slikovnog elementa. Spomenuti filtar je poznat pod engleskim nazivom *grayscale* ili skaliranje u sivo. U nastavku je navedena funkcija koju filtar obavlja nad slikovnim elementima

$$f(\ element(r,g,b)\ ) = element\ (\ (r+g+b)/3,\ (r+g+b)/3,\ (r+g+b)/3)$$

Primjer primjene filtra sivih razina prikazan je na slici 3.7., u lijevom djelu nalazi se originalna tekstura stabljike lješnjaka, dolje u sredini se nalazi rezultat filtriranja teksture, a desno se nalazi primjena preslikavanjem premještanjem geometrije.

Slika 3.7 - Prikaz pretvorbe slike u boji u visinsku mapu *grayscale* filtrom i primjena rezultata

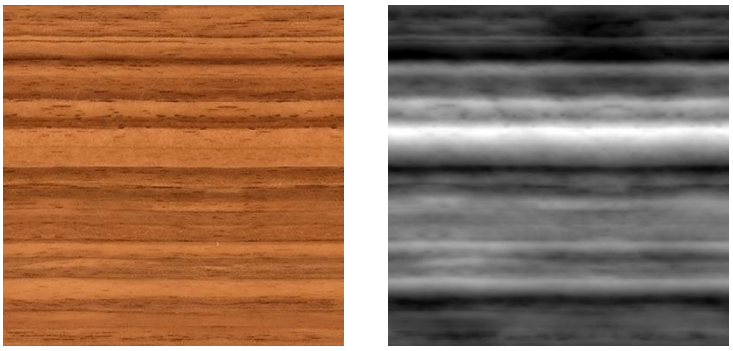
Iz desnog djela slike 3.7 se vidi da primjena spomenutog filtra daje relativno zadovoljavajuće rezultate i pristojnu visinsku mapu, no količina detalja na visinskoj mapi je ista kao i na izvornoj teksturi, koja u ovom slučaju nije imala područja visokih frekvencija, što neće biti općeniti slučaj. Ponekad je poželjno smanjiti frekvenciju alternacija vrijednosti slikovnih elemenata, jače naglasiti kontrast između „visokih“ i „niskih“ vrijednosti, smanjiti raspon ili promjeniti odnos između susjednih vrijednosti i slično. Profesionalniji alati za manipuliranje slikama poput programa Photoshop CS5 sadrže par načina za primjenu svih navedenih promjena i velik broj dodatnih mogućnosti.

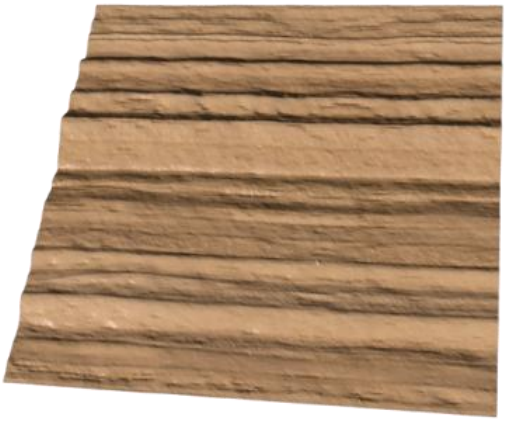
3.4.3.3 Programska konverzija

S obzirom na rastući broj računalnih igara, a time i profesionalnih umjetnika koji stvaraju modele i teksture koje čine izgled igre, raste i potreba za programima koji sami pretvaraju slike u boji u druge vrste tekstura za primjenu u igrama, a jedna od tih vrsta je i visinska mapa.

S obzirom na ne prevelik broj trenutno postojećih kvalitetnih alata, u nastavku se navodi jedan od najpoznatijih i najkorištenijih alata u industriji računalnih igara. Program *Crazy Bump* koji je izrađen 2006. godine sadrži velik broj prilagodljivih parametara za izradu tekstura koje se primjenjuju kao mape normala, visinske mape, mape reflektivne komponente i mape zaklanjanja okoline. Ulazna tekstura može biti tekstura boje, visinska mapa ili mapa normala. Navedeni program izvršava niz transformacija nad slikovnim elementima te stvara teksturu koja kvalitetno prenosi bitne detalje izvorne slike u rezultatntu.

Slika 3.8 prikazuje pretvorbu iste teksture stabljike lješnjaka u visinsku mapu, te primjenu rezultata, istim redoslijedom djelova kao u slici 3.7. Primjenjena tekstura izgleda neprirodnije i zaglađenije od rezultata *grayscale* konverzije isključivo zato što su odabrani takvi parametri pri pretvaranju izvorne teksture u samom *Crazy Bump* programu.



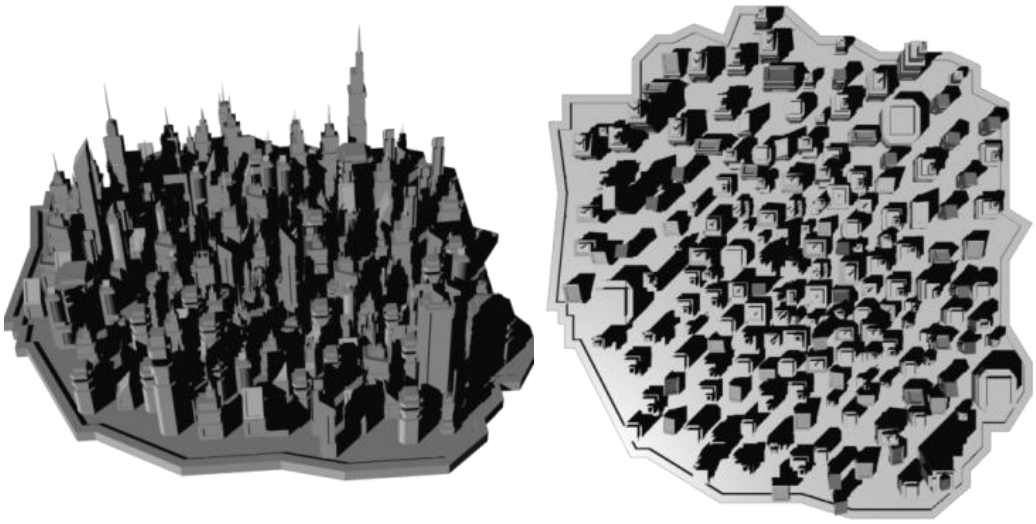


Slika 3.8 – Prikaz pretvorbe slike u boji u visinsku mapu programom Crazy Bump i prikaz rezultata

3.4.4. Izrada visinskih mapa na osnovi 3D modela objekata

Većina programa za općenito 3D modeliranje, poput programa 3ds Max, Maya, Blender, Modo i Zbrush, sadrži mogućnost preslikavanja modela u sceni na teksturu na više načina od projekcije na ravninu prikaza kamere kojom se promatra scena. Postupak izrade visinske mape na osnovi 3D modela u teoriji je jednak za sve programe, u praksi se neznatno razlikuje u mogućnostima, parametrima i naravno brzini izvođenja. U nastavku su koraci opisani u općenitom smislu, uz popratne slike za proces izrade visinske mape od modela u programu Modo.

Za početak, potrebno je učitati model od kojeg je cilj izraditi visinsku mapu. Slika 3.9 prikazuje učitan model grada od približno 50 tisuća trokuta. Zbog doživljaja perspektive, grad je prikazan iz tročetvrtinskog kuta (lijevo). Dodatno, zbog izrade visinske mape, grad je prikazan odozgo, s novim kutom osvjetljenja, koji nije bitan u ovom procesu izrade visinske mape.

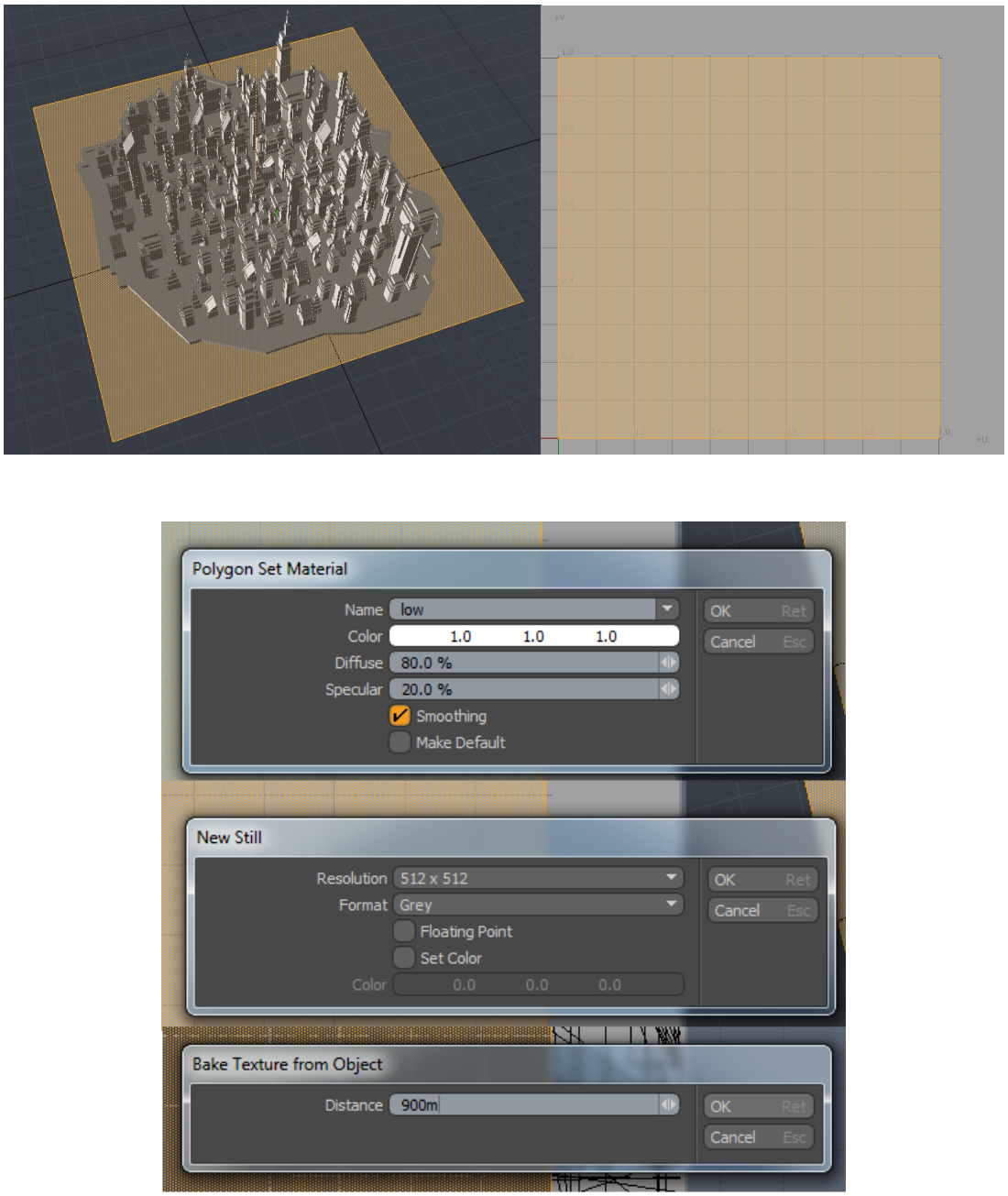


Slika 3.9 – Prikaz izvornog modela grada sa strane i okomito odozgo

Slijedeći korak je stvoriti model projektivne površine, iz kojeg će se projicirati zrake koje se trebaju sjeći s poligonima modela, kako bi rezultantna tekstura sadržavala visinsku mapu modela. Za model projektivne površine obično se koristi pravokutnik koji po veličini odgovara površini izvornog modela u podnoj ravnini (najčešće XZ ravnina, no ovisi o 3D programu). Model projektivne površine mora biti preslikan u UV prostor (Slika 3.10, drugi prikaz slijeva), kako bi rezultantna tekstura znala kamo pripada rezultat „bacanja zrake“ iz određene točke modela. U ovom slučaju, koristi se kvadratni model koji je preslikan na cijeli prvi kvadrant UV prostora. Pravokutni model se, ovisno o željenom rezultatu, postavlja ispod, iznad ili na sredini visine izvornog modela (Slika 3.10, drugi prikaz slijeva).

Nakon stvaranja (Slika 3.10, prvi prikaz slijeva) i ispravnog postavljanja projektivnog modela, dodjeljuje mu se materijal, u kojem se stvara prazna tekstura željene rezolucije (Slika 3.10, lijevi 2. red), u koju će biti pohranjen rezultat bacanja zrake. Slijedi određivanje parametra udaljenosti do koje će dosezati polupravac iz svake točke projektivnog modela pri pokušaju dohvaćanja točke poligona izvornog modela (Slika 3.10,

treći prikaz u 2. redu). Bitna činjenica je da se polupravci računaju i pokušavaju naći točke modela s obje strane projektivne površine.

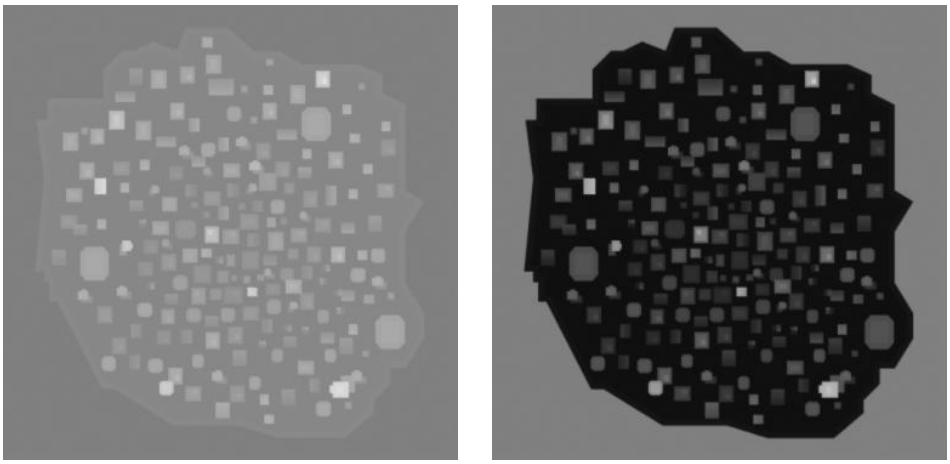


Slika 3.10 – Koraci namještanja scene i postavki za izradu visinske mape od modela postupkom „bacanja zrake“

Parametar udaljenosti bacanja zrake treba iznositi udaljenost u trenutnim jedinicama prostora 3D programa koja odgovara udaljenosti projektivnog modela do najdalje točke izvornog modela, te u ovom slučaju iznosi 900 metara. S obzirom da je projektivni model smješten ispod izvornog modela, pri izradi visinske mape pretpostavljena boja projektivne površine je neutralna visina, koja se nalazi na sredini intervala boja za prikaz visina, te iznosi 127, odnosno boju (127,127,127) – srednje sivu boju točno između bijele i crne.

Dodatno, budući da je projektivni model ispod izvornog modela, sve zrake koje su našle točke trokuta izvornog modela nalaze se s pozitivne strane projektivnog modela, te će svi slikovni elementi biti spomenute neutralno sive boje ili svjetlije, što u terminu visinskih mapa znači viša razina elevacije. Zaključuje se da je za bolje postizanje rezultata i korištenje cijelog raspona od 256 razina potrebno postaviti projektivni model na polovicu raspona visine izvornog modela. Budući da je sada manja udaljenost do krajnjih točaka, parametar duljine polupravca iznosi polovicu udaljenosti visinski najdaljih točaka izvornog modela, što za primjer iznosi 450 metara.

Slika 3.11 prikazuje rezultantne visinske mape od izvornog modela grada; za izradu lijevog prikaza projekcijski model se nalazio ispod modela, a u slučaju desnog prikaza na sredini visine modela.



Slika 3.11 – Prikaz visinskih mapa dobivenih na osnovi 3D modela metodom „bacanja zrake“

Što se tiče praktičnog dijela „bacanja zrake“, postupak je slijedeći:
Za svaki slikovni element rezultantne teksture određuje se prostorna koordinata trokuta projektivnog modela, koji odgovara položaju tog slikovnog elementa u prvom kvadrantu UV prostora u UV mapi tog modela. Dobivena prostorna koordinata je izvorišna točka „bacanja zrake“.

Za svaku izvorišnu točku određuje se prostorni segment pravca od točke koja iznosi T0 = (izvorišna točka + normala poligona projektivnog modela * parametar udaljenosti) do točke T1 = (izvorišna točka – normala * parametar udaljenosti). Polazeći od točke T0, promatra se postoji li sjecište segmenta pravca sa trokutom izvornog modela. Ukoliko sjecište postoji, na rezultantnu teksturu se pohranjuje slikovni element kojem su sve komponente boje (u slučaju visinske mape sa 256 razina) jednake omjeru udaljenosti točke sjecišta i T1 te udaljenosti točaka T0 i T1, u protivnom, slikovni element se boja predodređenom neutralnom bojom.

Potrebno je primjetiti kako je u primjeru površina iz koje se bacaju zrake savršeno planarna, što nije nužno u praksi, ali je najsmislenije pri izradi visinskih mapa od modela, za koje želimo da su planarni zapis udaljenosti točaka modela od ravne površine.

3.4.5. Dubinsko preslikavanje

U grafičkom protočnom sustavu postoji spremnik koji se zove dubinski spremnik (engl. depth buffer), te služi za pohranjivanje dubine svakog prikazanog slikovnog elementa scene. Nakon primjene matrica globalnog prostora, pogleda i projekcije na prostorne točke, one se preslikaju u 2D prostor te prikazuju na ekranu s obzirom na preslikane koordinate i promatrani okvir prostora (matrici pogleda). Uz projektivno preslikavanje,

transformirane točke sadrže i podatak o dubini, koji se zapisuje u dubinski spremnik. Ukoliko dvije ili više točaka imaju iste projekcijske (2D) koordinate, u spremniku boje ostaje zapisana ona koja ima najmanju dubinsku koordinatu, dakle ona točka koja je najbliže promatraču te zaklanja ostale slikovne elemente na tom položaju.

Prethodno navedena tehnika izrade visinske mape od modela „bacanjem zrake“ lagano je i efikasno ostvariva korištenjem sklopovlja i dubinskog spremnika. Potrebno je napomenuti da se u daljnjem opisu ne razmatraju mogućnosti sadržavanja točaka s transparentnom komponentom boje, iako se postupak ne razlikuje od navedenog slučaja. Postupak se ostvaruje slijedećim koracima:

Potrebno je učitavanje izvornog modela i definiranje matrice pogleda na model (najčešće pogled „odozgo“ iz pozitivne y osi prema negativnoj y osi), matrice transformacije modela, ako je potrebno dodatno skaliranje, rotiranje ili translacija, te matrice projekcije, za koju je poželjno da bude ortografska, a ne perspektivna, kako ne bi došlo do perspektivnog izobličenja ovisno o kutu pogleda i odnosu položaja modela i očišta. Primjena ortografske projekcije će skalirati prostorne točke te dati ispravne rezultate planarne udaljenosti točaka modela od očišta, koji će se zapisati u ispravnu visinsku mapu.

Pri izradi ortografske projektivne matrice potrebno je definirati volumen promatranog prostora pomoću šest brojeva, koji imaju značenje prostornih udaljenosti stranica kvadra od točke položaja očišta, što je zaseban podatak. Dvije udaljenosti koje utječu na dubinski spremnik su bliska ravnina i daleka ravnina (engl. *near plane, far plane*). Sve prostorne točke modela čija se dubinska koordinata nalazi između navedenih vrijednosti ravnina će biti razmatrane kao kandidati da se upišu u spremnik boje i dubinski spremnik, te se one najbliže promatraču konačno iscrtaju na zaslonu.

Nakon definiranja potrebnih matrica i učitavanja modela, potrebno je primjeniti sve matrice na svaku točku modela, te pročitati dubinsku koordinatu transformirane točke. Ovisno o tehnologiji, moguće je postići da grafički protočni sustav obavi sve navedene transformacije i pohranu u dubinski spremnik. Preostali dio je pročitati dubinski spremnik, te s obzirom na ranije definiranu blisku i daleku ravninu, kvantizirati dubinu svake točke u željeni raspon vrijednosti elevacija rezultantne visinske mape.

U nastavku je navedena formula pretvorbe dubine točke na određenom položaju rezultantnog prikaza scene u boju slikovnog elementa visinske mape, pod pretpostavkom da je broj različitih intenziteta manji ili jednak 256:

Parametri: *near* – udaljenost bliske ravnine, *far* – udaljenost daleke ravnine, *depth(x,y)* – dubinska vrijednost točke na koordinatama x i y zapisane u dubinskom spremniku nakon obrade cijele scene, *intensity_count* – broj intenziteta koje želimo razlikovati u visinskoj mapi (pretpostavka da je najniži intenzitet na vrijednosti 0), *x i y* – koordinate slikovnog elementa scene i visinske mape

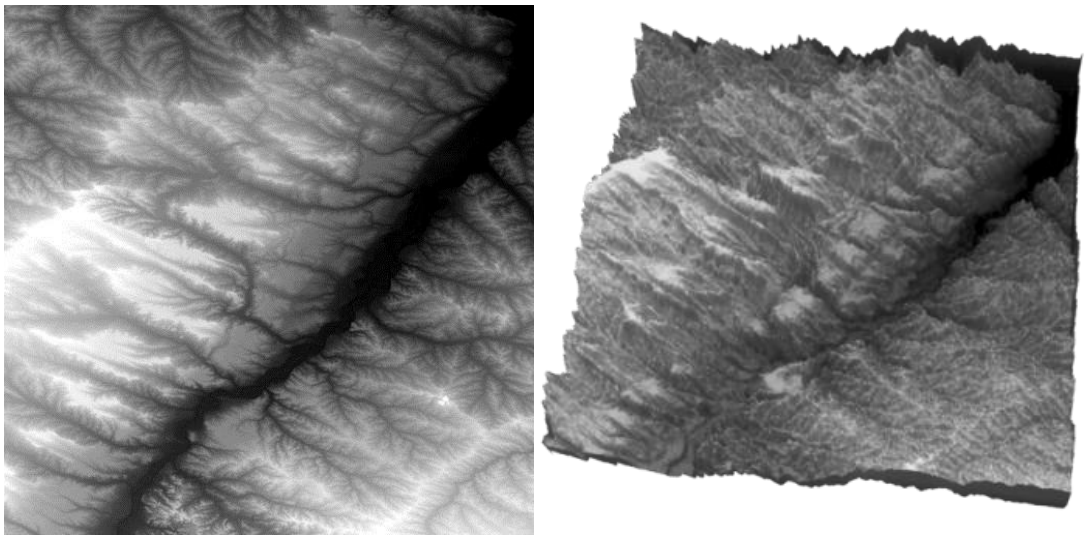
*Color(x,y) = (far - depth(x,y)) / (far – near) * (intensity_count - 1)*

Ukoliko se koristi model grada iz prethodnih primjera, te se paramtetri bliske i daleke ravnine postave na vrijednosti -450 i 450 redom, rezultantna visinska mapa bi izgledala približno identično visinskoj mapi dobivenoj tehnikom „bacanja zrake“.

3.4.6. Preuzimanje gotovih visinskih mapa s Interneta

Najbrži način dolaska u posjed visinskih mapa je pohranjivanje tekstura dohvaćenih s Interneta u memoriju računala. Postoje profesionalne i znanstvene stranice koje sadrže razne vrste kvalitetnih tekstura koje su već pripremljene da se primjene kao visinske mape. Neke stranice sadrže obrađene fotografije različitih prirodnih i umjetnih površinskih materijala, a neke sadrže slike zemljine površine slikane pomoću satelita. U referencama postoje linkovi na korisne stranice navedene prirode. Na slici 3.12 prikazana je tekstura skinuta s interneta u prikazanom obliku i izravno primjenjena u

preslikavanju pomicanjem geometrije, te je ista tekstura primjenjena za modifikaciju boje. Slika preuzeta sa stranice [7] je visinski prikaz Zemljine površine na području između 47 i 48 stupnjeva geografske visine te 104 i 105 stupnjeva geografske širine, konkretnije iz sjevero-istočnog djela države Montane u SAD-u.



Slika 3.11 – Prikaz i primjena satelitske mape Zemljine površine u formatu zapisa identičnom visinskoj mapi

3.5. Primjene visinskih mapa

Stvaranje visinskih mapa je moguće u bilo kojem programu za stvaranje bojanje tekstura, no budući da efekt primjene visinske mape uglavnom nije očit iz njenog prikaza, postoji puno programskih aplikacija koje u realnom vremenu vizualiziraju primjenu visinske mape na model ili na proces stvaranja modela. U takvim aplikacijama visinske mape se mogu uređivati izravno na modelima primjenom raznih alata koji povisuju, snižavaju, izravnavaju ili erodiraju teren, te ga alterniraju na način na koji razne prirodne pojave modificiraju površinu u prirodi, a sve što spomenuti alati rade je modificiranje slikovnih elemenata dvodimenzionalne teksture, ili interpretacije istih.

Načini primjene

Visinska mapa se može iskoristiti u više svrha, no sve se svode na tri osnovne primjene koje će biti obrađene u ovom poglavlju i par drugih primjena koje funkcioniraju u suradnji s mapama normala te će biti obrađene zajedno sa primjenama mapa normala. Osnovne kategorije primjena gdje su aktori isključivo visinske mape su slijedeće:

1. preslikavanje neravnina (engl. bump mapping) u kojem se tekstura koristi za stvaranje detaljnijih sjena na površini,
2. preslikavanje premještanjem geometrije (engl. displacement mapping) u kojem se stvarnoj geometriji modela mijenja položaj na temelju postojećih normala i boje sa teksture
3. generiranje 3D modela na temelju rezolucije i boja tekture

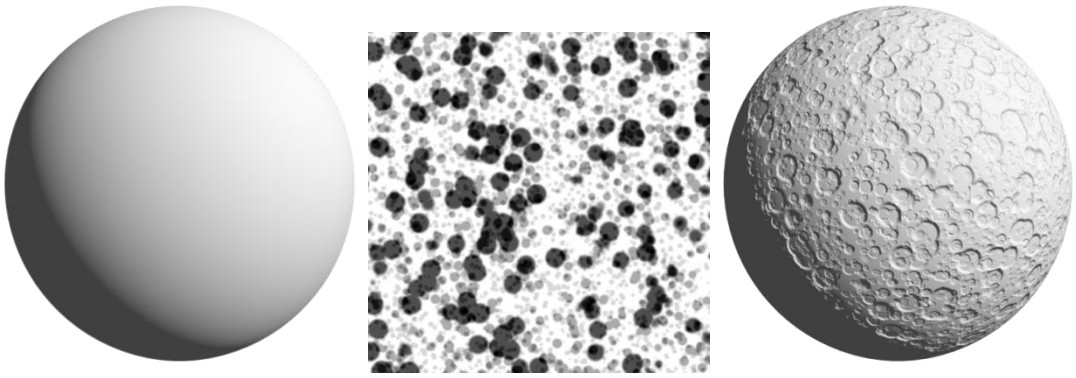
3.5.1. Preslikavanje neravnina

Preslikavanje neravnina (engl. *bump mapping*) je postupak u kojem se posebna tekstura primjenjuje na model kako bi se simuliralo postojanje nabora i neravnina na površini modela. To se postiže modulacijom normala površine objekta te korištenjem promjenjenih normala pri izračunavanju iznosa osvjetljenja modela, te se najčešće s Phongovim modelom s intenzitetom reflektivne komponente (engl. *Phong reflection model*). Vizualni

rezultat je naizgled naborana površina nastala od ravnih, glatkih poligona, iako se broj poligona i pravi izgled modela nije promijenio, zbog čega je ova primjena daleko brža i manje zahtjevna od metode premještanja geometrije.

U prethodnom paragrafu je napomenuto da se za preslikavanje neravnina koristi posebna tekstura, gdje je primjena visinske mape samo jedan od načina, te će on biti razmatran u ovom poglavlju. Drugi, najčešći način ostvarivanja preslikavanja neravnina je pomoću mape normala, te se razmatra u idućim poglavljima.

Lijevi dio slike 3.12 prikazuje model kugle, čiji prikaz je ostvaren primjenom Phongovog modela osvjetljenja i sjenčanja, bez primjene tekstura. U sredini slike nalazi se visinska mapa, a desno rezultat primjene visinske mape na preslikavanje neravnina površine modela kugle, na čiji rezultat bitno uječe broj, položaj i vrsta svjetla u sceni. U ovom slučaju, u sceni postoji samo jedno simulirano „beskonačno udaljeno“ usmjereno svjetlo.



Slika 3.12 Prikaz osnovne primjene teksture na preslikavanje neravnina

Kao što je objašnjeno u prethodnom poglavlju, dio podataka modela su vektori normale na svaku točku modela. U grafičkom protočnom sustavu podaci o vektorima normala interpoliraju između točaka (engl. *vertex*) te u instrukcije za sjenčanje svakog vidljivog slikovnog elementa se šalje rezultat interpolacije normala, između ostalih podataka, za poligon modela na kojem se taj slikovni element nalazi. Ukoliko se informacija o normali izravno upotrijebi u izračunu modela osvjetlljenja i sjenčanja kako bi se dobila rezultantna boja, dobiva se izgled poput lijevog prikaza na slici 3.12. Glavni

element funkcionalnosti tehnike preslikavanja neravnina je modifikacija normale za svaki slikovni element ovisno o položaju UV koordinata tog slikovnog elementa u UV mapi modela.

U samom ostvarenju skupa instrukcija programa za sjenčanje slikovnih elemenata postoje varijacije izvedbi ove tehnike. Najčešći način primjene podataka visinske mape je opisan u nastavku, uz priložen pseudokod i pretpostavku postojanja korištenih funkcija.

Ulazni podaci (*cjelobrojne koordinate trenutnog slikovnog elementa na UV prostoru modela (UV.x, UV.y), decimalni zapis 3D vektora normale trenutnog slikovnog elementa interpolirana između susjednih točaka modela – UlaznaNormala(x,y,z)*)

1. Za trenutni slikovni element scene dohvati okolne slikovne elemente visinske mape na temelju trenutnih UV koordinata. Pod pretpostavkom najčešćeg slučaja, dohvaćaju se prvi susjedni slikovni elementi. S obzirom da su u bojama visinske mape iznosi intenziteta pojedinih kanala jednaki, uzima se vrijednost crvenog kanala.

IntenzitetGore = DohvatiBoju(visinskaMapa, UV.x,UV.y - 1).crvena;
IntenzitetDolje = DohvatiBoju(visinskaMapa, UV.x,UV.y + 1).crvena;
IntenzitetLijevo = DohvatiBoju(visinskaMapa, UV.x - 1,UV.y).crvena;
IntenzitetDesno = DohvatiBoju(visinskaMapa, UV.x + 1,UV.y).crvena;

2. Računa se razlika intenziteta lijevog i desnog, te gornjeg i donjeg elementa, te se razlike pohrane kao vektori visinskih promjena u prostoru oko trenutnog ciljnog elementa. U ovom slučaju vektori su orijentirani kao rezultantna slika na ekranu, vrijednosti x osi rastu prema desno, vrijednosti y osi prema gore, a z osi prema promatraču, iz ekrana.

HorizontalnaRazlika = IntenzitetDesno – IntenzitetLijevo;
VertikalnaRazlika = IntenzitetGore – IntenzitetDolje;

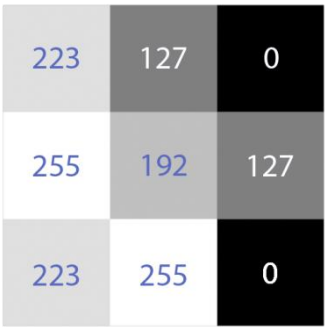
VektorHorizontalnePromjene = (255,0,HorizontalnaRazlika);
VektorVertikalnePromjene = (0,255,VertikalnaRazlika);

3. Konačna normala se dobiva normalizacijom zbroja izvorne normale slikovnog elementa i normaliziranog vektorskog produkta vektora visinskih promjena.

NormalaIzVisinskeMape = VektorskiUmnožak(VektorHorizontalnePromjene, VektorVertikalnePromjene);
NormalaIzVisinskeMape = Normaliziraj(NormalaIzVisinskeMape);

KonačnaNormala = Normaliziraj(UlaznaNormala + NormalaIzVisinskeMape);
Slijedi konkretan primjer vrijednosti visinskih intenziteta i popratne slike.

Pretpostavka je da trenutni slikovni element i njegovi susjedi imaju inzenzitete boja kako je prikazano brojevima na slici 3.13, što odgovara bojama u pozadini, budući da su intenziteti jednaki u sva tri kanala boje, one pripadaju crno-bijelom gradijentu.



Slika 3.13 – Prikaz intenziteta i boja kao djela visinske mape analiziranog u primjeru

Vrijednosti intenziteta, njihovih razlika i vektori promjene za brojeve iz primjera iznose:
IntenzitetGore = 127;
IntenzitetDolje = 255;
IntenzitetLijevo = 255;
IntenzitetDesno = 127;

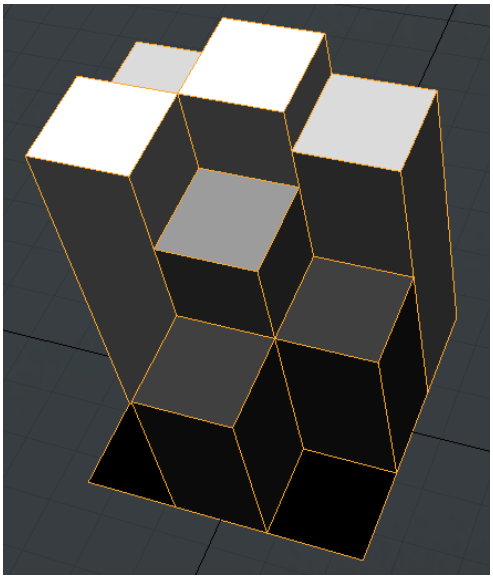
HorizontalnaRazlika = -128;
VertikalnaRazlika = -128;

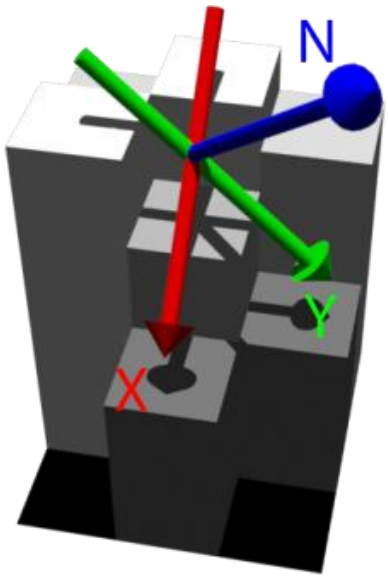
VektorHorizontalnePromjene = (255,0,-128);
VektorVertikalnePromjene = (0,255,-128);

Intenzitete možemo vizualizirati prostornim kvadrima, jednake baze i visine jednake intenzitetu slikovnog elementa, kako je prikazano lijevo u slici 3.14. Na desnom dijelu iste slike nalazi se vizualizacija vektora promjere (oznake: X za horizontalni, Y za vertikalni), te vektora koji se njihovim vektorskim produktom (označen slovom N). Model stupaca koji prikazuju intenzitete je prikazan iz različitog kuta od prikaza slikovnih elemenata sa slike 3.13 zbog bolje preglednosti.

Pod pretpostavkom da interpolirana vrijednost normale trenutnog slikovnog elementa iznosi (0,0,1), slijedi konačni račun primjera:
NormalaIzVisinskeMape = Normaliziraj [(255, 0, -128) x (0, 255, -128)];
NormalaIzVisinskeMape = (0.4093, 0.4093, 0.8154);

KonačnaNormala = Normaliziraj((0,0,1) + (0.4093, 0.4093, 0.8154));
KonačnaNormala = (0.2148, 0.2148, 0.9528);



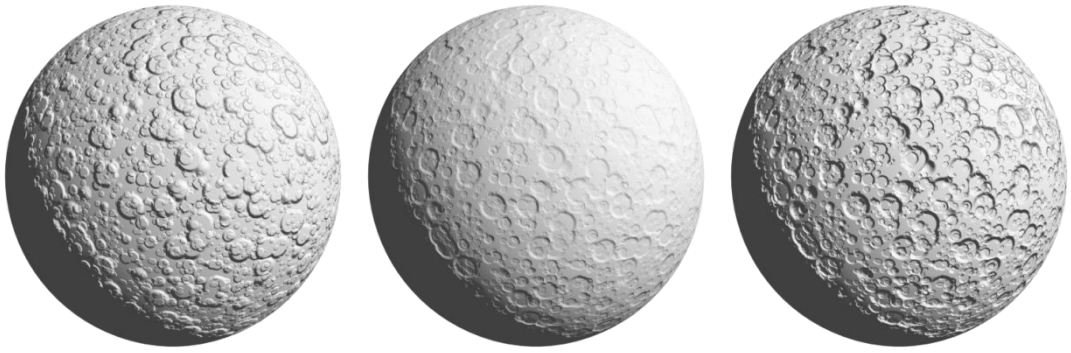


Slika 3.14 – Vizualizacija intenziteta, vektora promjene i rezultata

Nakon izračunavanja nove normale slikovnog elementa, ona se koristi u određivanju osvjetljenja tog elementa. Konkretno, u izračunu intenziteta difuzne komponente normala se skalarno množi s vektorom upada svjetlosti na točku, a u izračunu intenziteta reflektivne komponente, vektor upada svjetlosti se zrcali s obzirom na normalu prije daljnjih proračuna.

Budući da je zapis intenziteta visinske mape kvantiziran, javlja se potreba za preciznijim manipuliranjem smjera rezultantnog vektora, te se ostvaruje skaliranjem razlika intenziteta prije računanje vektorskog umnoška vektora promjena. Na slici 3.15 prikazana je primjena visinske mape s prikaza 3.12 za različite vrijednosti parametra skaliranja razlika. Parametar skaliranja utječe na rezultantni vektor na slijedeći način:

Ukoliko se istim parametrom množe i horizontalna i vertikalna razlika, ako je parametar bliže nuli, normala će biti bliža vektoru (0,0,1), a što je parametar veći, normala će težiti vektoru (1,1,0). Kad bi se razlike množile različitim parametrima, njihova promjena bi na konačan prikaz scene imala kombinirani efekt promjene intenziteta i rotacije svjetla u prostoru scene u odnosu na model.



Slika 3.15 – Prikaz preslikavanja neravnina s vrijednostima parametra skaliranja (-15 mm, 5 mm, 25 mm) na modelu kugle promjera 1 metar

3.5.2. Preslikavanje neravnina premještanjem geometrije

Kao kontrast metodi preslikavanja neravnina, koja je na temelju visinske mape alternirala normale površine, no nije mijenjala samu geometriju, razvula se metoda preslikavanja premještanjem geometrije (engl. *displacement mapping*). Ova metoda koristi postojeće pohranjene podatke o normalama točaka modela, te ih koristi kako bi na temelju intenziteta slikovnih elemenata visinske mape promijenila položaje točaka u prostoru scene.

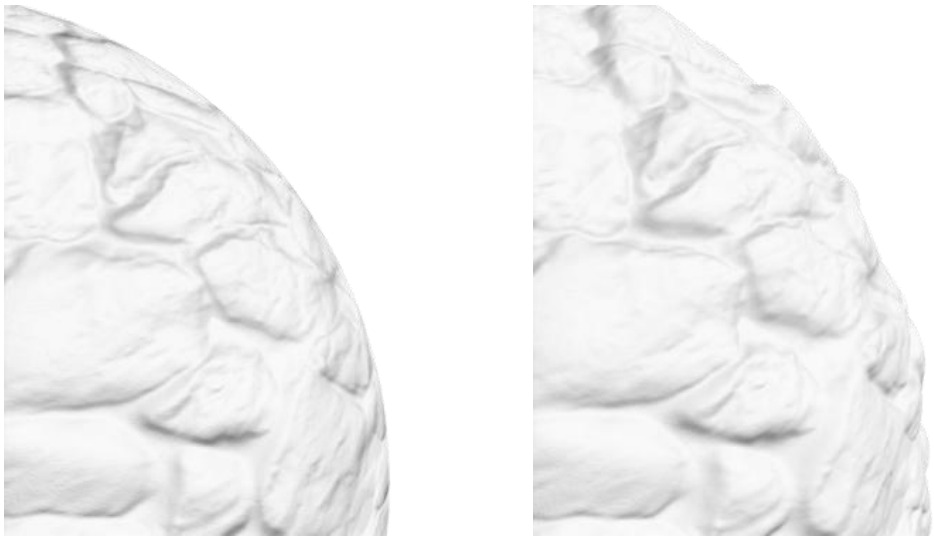
Prije nastavka, potrebno je napomenuti da tri načina implementacije preslikavanja premještanjem geometrije, koji se temelje na tri postojeće vrste programa za sjenčanje za koje je moguće pisati instrukcije, te ih slati procesorima grafičkih kartica, a to su: programi za sjenčanje točaka (engl. *vertex shader*), geometrije (engl. *geometry shader*), i slikovnih elemenata (engl. *fragment shader* ili *pixel shader*).
Budući da je cilj ove metode povećati razinu detalja u konačnom prikazu modela, a ne procesuirati veću količinu točaka u procesoru i slati ih u grafički protočni sustav, u nastavku poglavlja je naglasak na način implementacije koji koristi programe za sjenčanje geometrije.

Za povećanje razine detalja, sama promjena položaja postojećih točaka modela nije dovoljna, zbog čega se stvaraju nove točke između postojećih točaka, koje se također pomiču na isti način, te se tako dobije na količini vizualnih detalja. Kako se razvijao grafički protočni sustav, dugo nije postojala mogućnost stvaranja novih točaka na grafičkoj kartici, već su svi podaci stvarani na procesoru, a zatim slani protočnom sustavu na obradu. Zato su raniji visoko kvalitetni sustavi za računanje prikaza slike (engl. *rendering system*) ostvarivali metodu preslikavanja premještanjem geometrije na procesoru, dok to nije omogućeno razvojem programskih sučelja (poput OpenGl-a i DirectX-a) i grafičkih kartica sa istim mogućnostima. Danas se koriste prethodno spomenuti programi za sjenčanje geometrije, u kojima je moguće stvarati nove točke na temelju postojećih točaka, ili s potpuno novim, nezavisnim podacima, i to u protočnom sustavu. Budući da nastaju nove točke, nastaju novi poligoni, koji mogu zaklanjati ili bacati sjenu na druge poligone, a moguć je i nastanak potpuno drukčijeg modela, tako da ova

metoda ne simulira veliku količinu detalja promjenom normala, već ju stvara dodavanjem geometrije i promjenom normala i položaja svih točaka.

Uz postojanje metode premještanja geometrije preslikavanjem, čije ime naglašava da u njenoj primjeni postoji tekstura koja je izvor temeljnih podataka za primjenu metode, postoji i metoda premještanja koja koristi proceduralno stvorene teksture i uzorke, u kojoj se podaci temelje na parametrima algoritma, te nije potrebno preslikavanje točaka modela u UV prostor, ni slanje teksture visinske mape u protočni sustav. Zbog jasnoće terminologije, pojam metoda premještanja geometrije se koristi kao koncept koji obuhvaća sve navedene metode.

Slika 3.16 prikazuje primjenu visinske mape metodom preslikavanja neravnina (lijevo) i metodom pomaka geometrije (desno) kako bi ilustrirala osnovnu razliku između njih. Budući da prva samo mijenja detalje, silueta ovog dijela kugle je i dalje zaglađena, jer točke nisu promijenile položaj. Druga dodaje nove točke, te je silueta nezaglađena, što odgovara sjenčanom modelu kamene površine nastalog od modela kugle stvaranjem novih točaka koje u protočnom sustavu tvore nove površine pri računanju izgleda konačne slike.



Slika 3.16 – Usporedba metode preslikavanja neravnina (lijevo) i premještanja geometrije (desno)

Budući da metoda premještanja geometrije preslikavanjem u programu za sjenčanje geometrije stvara nove točke na temelju podataka

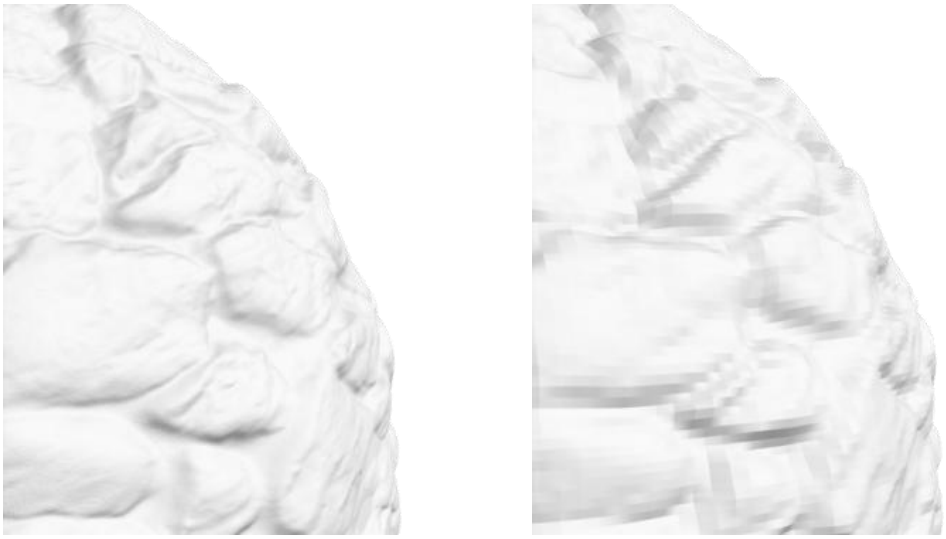
visinske mape, broj stvorenih točaka ovisi o samom preslikavanju modela u UV prostor, rezoluciji visinske mape i preslikavanju visinske mape u UV prostoru. U nastavku je dan opis postupka preslikavanja premještanjem geometrije.

Prilikom obrade svakog trokuta modela u protočnom sustavu, za njega se odrede koordinate UV mape modela koje on obuhvaća, te se za svaki par koordinata (x,y) dohvati slikovni element visinske mape i izračuna interpolacijska normala na temelju normala točaka koje čine taj trokut, zajedno s interpolacijom položaja točkaka. Dodatno, interpoliranjem se izračunaju i ostali podaci, poput vektora tangente lokalnog prostora te točke, vektora upada svjetlosti, boje pridružene točci i slično.

Nakon pripreme navedenih podataka, računaju se nove prostorne koordinate za svaku novu točku, a vrijednosti ostalih podataka se preuzimaju iz interpoliranih podataka. Novi položaj točke (p') jednak je zbroju položaja interpolirane točke na trokutu (p) i umnošku interpolirane normale ($\vec{n}$), intenziteta slikovnog elementa visinske mape (h) i opcionalnog faktora skaliranja intenziteta (f_s), prisutnog i u preslikavanju neravnina. Matematički izraz glasi:

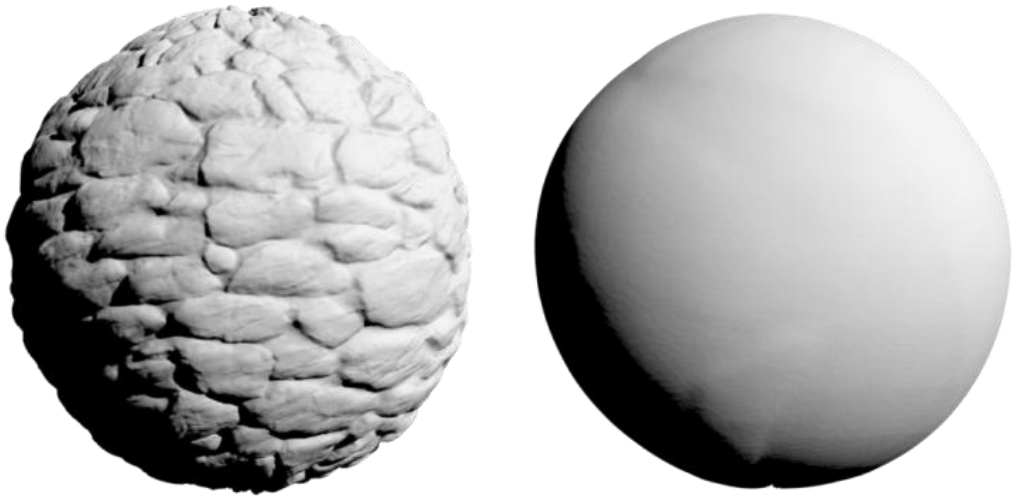
$$p'(x,y,z) = p(x,y,z) + \vec{n}(x,y,z) * h(u,v) * f_s$$

Spomenuto je da količina novih točaka, a time i vizualnih detalja, ovisi o rezoluciji visinske mape. Slika 3.17 prikazuje rezultate primjene dvaju tekstura istog izvornog sadržaja, no lijeva tekstura je rezolucije 2048x2048 slikovnih elemenata, a desna samo 256x256 elemenata. Za ovu sliku, oba prikaza su rađena na istom modelu, sa istom UV mapom.



Slika 3.17 – Prikaz utjecaja rezolucije visinske mape na kvalitetu prikaza u primjeni metode premještanja geometrije

Kao što je objašnjeno u opisu postupka, broj točaka ovisi o broju slikovnih elemenata visinske mape koje prekriva pojedini trokut. Slika 3.17 je ilustrirala slučaj u kojem trokut prekriva manje slikovnih elemenata jer je rezolucija desne teksture bila manja. Slika 3.18 pokazuje utjecaj preslikavanja modela na UV prostor. Na lijevoj strani je prikazan izračun scene u kojoj je model kugle preslikan na način da ispunjava većinu UV prostora. Na desnoj strani je scena sa modelom kugle koji je preslikan na kvadrat UV prostora koji je omeđen točkama (0,0) i (0.125,0.125), točno 64 dio površine osnovnog kvadranta UV prostora. Za visinsku mapu oba modela korištena je ista tekstura, korištena na primjeru desnog prikaza slike 3.17.



Slika 3.18 – Prikaz utjecaja UV mape modela na kvalitetu prikaza u primjeni metode premještanja geometrije

Iz specifikacija UV mapa modela jasno je da će model iz desnog prikaza biti preslikan na samo dio slikovnih elemenata teksture, te će taj dio, budući da sadrži manje elemenata, stvoriti manje novih točaka. Desni prikaz nije neispravan, nego je rezultat neusklađivanja UV preslikavanja i preslikavanja teksture u UV prostoru. Kad bi tekstura pri računanju izgleda modela iz desnog prikaza bila preslikana na isti kvadrat UV prostora kao i model, rezultat bi izgledao isto poput lijevog prikaza na slici 3.18.

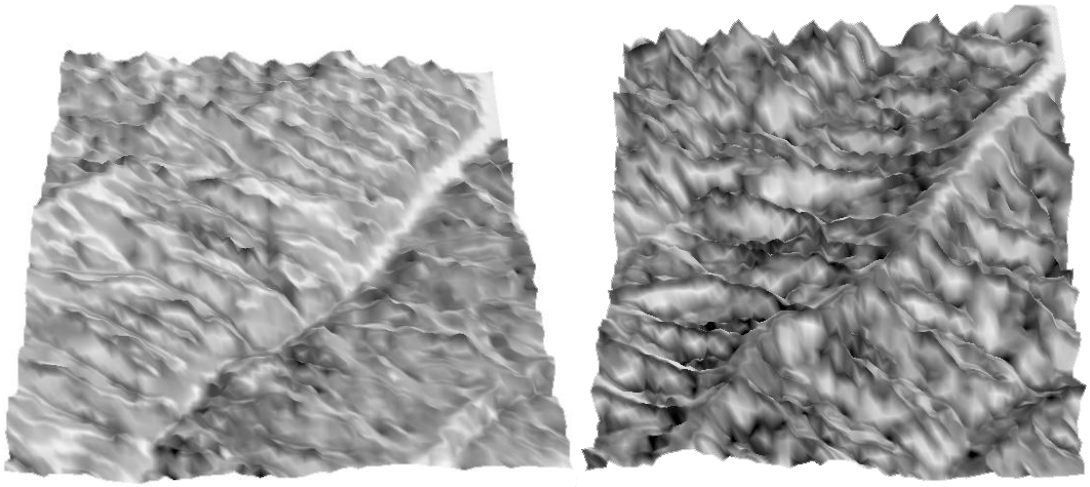
Ova tehnika pruža pravo stvaranje detaljne površine u grafičkom protočnom sustavu, te se u praksi koristi zajedno s tehnikama adaptivne teselacije, koja se koristi u simulacijama i računalnim igrama. Tehnike adaptivne teselacije se temelje na dinamičkoj podjeli modela i stvaranju novih točaka iz visinske mape, gdje sama količina ili grubost podjele ovisi o udaljenosti očišta i pojedinih djelova modela.

Drugi spomenuti način implementacije metode korištenja visinske mape premještanjem geometrije je programiranje u programu za sjenčanje točaka. Taj pristup, za razliku od dosad obrađivanog pristupa s programom za sjenčanje geometrije, mora kao ulazne podatke primiti model visoke rezolucije, koji ima sve potrebne točke, zato što grafički protočni sustav ne dopušta stvaranje novih točaka u dijelu koji izvršava instrukcije programa za sjenčanje točaka. Primjena na sve poslane točke je ista kao u prošlom načinu, računa se novi položaj točaka s obzirom na stari položaj, normalu točke i intenzitet s odgovarajućeg slikovnog elementa visinske mape.

Ovaj pristup zahtjeva da ulazni model bude sastavljen od kvalitetno raspoređenog velikog broja točaka, te je ograničen na modificiranje udaljenosti. U ovom pristupu ne postoji mogućnost dodavanja geometrije i detalji se ne stvaraju, nego se isključivo pomiču točke modela, što se u praksi radi u programu za modeliranje, a ne programu za sjenčanje i dodavanje detalja gotovom modelu. Najmodernije arhitekture protočnog sustava dopuštaju dohvaćanje tekstura u programu za sjenčanje točaka, ali velika većina računala u upotrebi još nema komponente s tom funkcionalnošću. Uz

navedeno, samo dohvaćanje teksture je puno sporije u dijelu za sjenčanje točaka nego u dijelu za sjenčanje slikovnih elemenata. Konačno, dio instrukcija u programu za sjenčanje točaka se izvršava za sve poslane točke, što može znatno usporiti prikaz ako se dodaju instrukcije koje nisu prilagođene tom dijelu protočnog sustava. Sve navedeno su razlozi zašto se ovaj način implemetacije rijetko koristi u praksi.

Slika 3.19 prikazuje model kvadrata veličine 5x5 metara, sastavljenog od 64 horizontalna i 64 vertikalna kvadrata, 6225 točaka postavljenih u pravilnu mrežu u ravnini XY osi. Oba prikaza predstavljaju primjenu načina implementacije metode korištenja visinske mape premještanjem geometrije programiranjem u programu za sjenčanje točaka. Lijevi prikaz ima vrijednost parametra skaliranja 15, desni prikaz vrijednost -30. Tekstura primjenjena kao visinska mapa je ista tekstura kao u lijevom prikazu na slici 3.11.



Slika 3.19 – Prikaz primjene premještanja geometrije u programu za sjenčanje točaka na detaljno podijeljeni model većeg broja točaka

Treći način implementacije je u programu za sjenčanje slikovnih elemenata. Za razliku od prethodna dva načina, ova implementacija ne mijenja položaje točaka modela, niti ih stvara, te se zato naziva metodom virtualnog premještanja geometrije. Budući da se implementacija radi u programu za sjenčanje slikovnih elemenata, dostupni podaci su rezultati interpolacije i točno određeni za svaki slikovni element, te jedini utjecaj koji kod može imati je na konačnu boju i dubinu svakog elementa.

Postoji velik broj metoda za samo ovaj način implementacije. Skupni naziv tih metoda je preslikavanje zaklanjanja uslijed paralakse ili promjene pogleda (engl. *parallax mapping*). S obzirom da se kod ovih tehnika u pravilu pomiču samo elementi teksture u ravnini poligona, bolji naziv bi bio tehnika pomicanja elemenata teksture. Bez ulaženja u detalje implementacije pojedine metode, nastavak ovog podpoglavlja se koncentrira na općenitu tehniku pomicanja elemenata teksture.

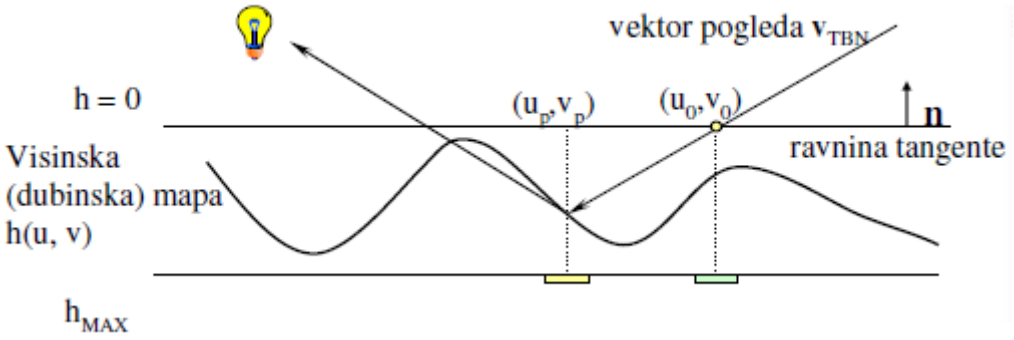
Izvedba tehnike se temelji na određivanju slikovnog elementa koji je vidljiv iz očišta, s obzirom na podatke iz visinske mape koji u ovoj metodi virtualno „pretvaraju“ površinu modela u neravnu površinu. Konkretno, problematika ovog pristupa je pronalaženje UV koordinata prvog slikovnog elementa koji se nalazi na sjecištu zrake između očišta i trenutnog slikovnog elementa u programu za sjenčanje elemenata.

Slijedi opis općenitog postupka za traženje koordinata slikovnog elementa na sjecištu zrake iz očišta. Najbitniji poznati podaci su: vektor u tangencijalnom prostoru iz očišta prema prostornoj točci koja odgovara trenutnom slikovnom elementu (označen kao vektor pogleda na slici 3.20) i trenutne UV koordinate na UV mapi modela (označene žutim pravokutnikom i koordinatama (u_p, v_p) na slici 3.20).

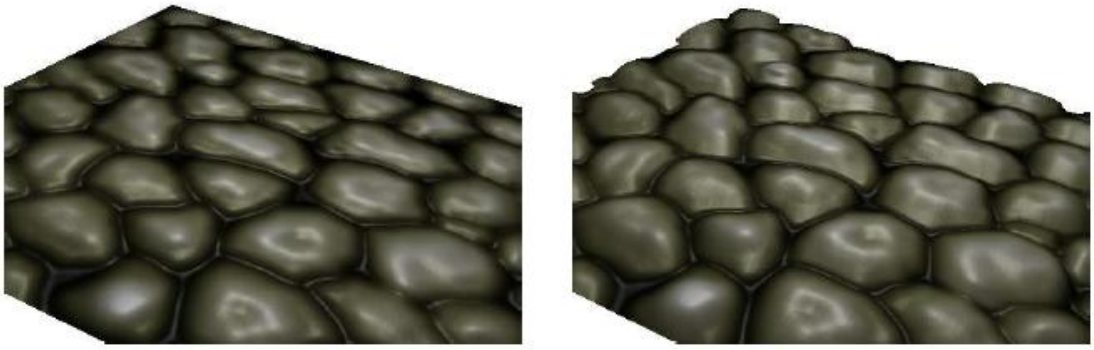
Pretpostavka je da zraka iz očišta neprekinuto pada na slikovni element danih koordinata, ili presijeca virtualnu površinu, koja je rezultat pomaka slikovnih elemenata na temelju intenziteta visinske mape, u području koje je bliže promatraču od promatranih UV koordinata. Zbog navedene pretpostavke, potrebno je odrediti projekciju vektora prema promatraču i kretati se po UV prostoru sve do nalaska zadovoljavajućeg sjecišta zrake i slikovnog elementa.

Do zadovoljavajućeg sjecišta može se doći kretanjem za iznos spomenutog projiciranog vektora od trenutnih UV koordinata prema očištu, ili naći pretpostavljeno prvo sjecište neravne virtualne ravnine (označene zelenim pravokutnikom i koordinatama (u_0, v_0) na slici 3.20) i zrake iz očišta te se kretati prema UV koordinatama trenutnog slikovnog elementa.

U oba slučaja za svaki korak postupka traženja sjecišta potrebno je učitavati novi intenzitet visinske mape za nove trenutne UV koordinate. Kad se nađe zadovoljavajuća točka presjeka, rezultantne UV koordinate se koriste za dohvaćanje elementa teksture, koji se potom koriste za izračun intenziteta osvjetljenja i konačne boje slikovnog elementa prikaza scene čiji podaci su došli na ulaz programa za sjenčanje elemenata.



Slika 3.20 ilustrira situaciju koju predstavljaju podaci u programu za sjenčanje slikovnih elemenata pri izračunu preslikavanja zaklanjanja uslijed paralakse.



Slika 3.21 Primjene visinske mape na model preslikavanjem neravnina (lijevo) i implementacijom preslikavanja zaklanjanja uslijed paralakse (desno) [13]

U metodi preslikavanja zaklanjanja moguće je implementirati ispravne sjene na teksturi uzrokovane „virtualno“ uzdignutim slikovnim elementima na temelju visinske mape, te prikaz siluete ruba teksture ovisno o visinama, kao što je vidljivo na desnom prikazu slike 3.21. Zbog dodavanja iznimne razine detalja na visokom stupnju realizma bez promjene položaja ili broja točaka, visokih peformansi, jednostavnoj implementaciji i velikoj podržanosti grafičke opreme, implementacije ove metode su sve popularnije za primjenu u

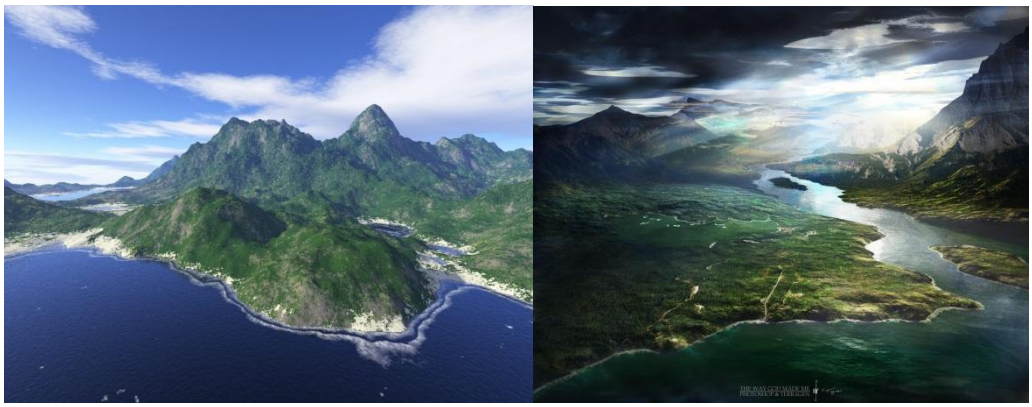
računalnim igrama, simulacijama letova, biološkim simulacijama i drugim vrstama aplikacija koje prikazuju 3D scene u realnom vremenu.

3.5.3. Generiranje novih 3D modela i simulacija na temelju visinske mape

Do sada su razmatrani načini primjene visinskih mapa za povećanje detalja i poboljšanje realizma u prikazu modela,uglavnom s rezultatima u realnom vremenu. Generiranje 3D modela na temelju visinskih mapa je blisko metodi premještanja geometrije u programu za sjenčanje geometrije, no u toj metodi nove točke se generiraju na temelju podataka postojećih točaka. U slučaju korištenja visinskih mapa za stvaranje modela i simulacija, ulazni parametri su sama mapa i skupina parametara, no nema postojećih podataka, već se svi podaci generiraju na temelju parametara i kvantiziranih vrijednosti visinske mape.

Najpoznatija i najočitija upotreba visinskih mapa je izrada 3D modela terena koji se dalje koriste u računalnim igrama, simulacijama ili kompoziciju scene za teksturu u umjetničke svrhe.

Jedni od najpoznatijih programa za izradu prikaza 3D terena i njegovih animacija su Vue i Terragen. Slike 3.22 i 3.23 prikazuju primjenu visinskih mapa za stvaranje terena i površinskih neravnina u izradi prikaza scene.



Slika 3.22 – Prikaz scena izrađenih u programu Terragen [15] [16]

Slika 3.23 – Prikaz scena izrađenih u programu Vue [17] [18]

Jedna od izraženijih simulacija prikaza terena su neki preglednici postojećih 3D terena izrađenih na temelju satelitskih snimaka Zemljine površine, koji primjenom visinskih mapa postižu dodatni realizam u prikazima.

Druge vrste simulacija i manje očiti načini korištenja koji iz visinskih mapa izvode podatke i zaključke na temelju kojih simuliraju ponašanje ili virtualno okruženje su: 3D planiranje letova, geomorfološke analize terena, simulacije leta i simulacije ponašanja sustava čestica (poput toka vode, lavina i odrona zemlje), inženjerske analize za dizajn infrastrukture građevina, analize i predviđanja arheoloških iskopina, analize sustava za asistiranje upravljanjem prijevoznim sredstvima i slično.

Sve prethodno navedene primjene su praktične i korisne, no zbog izvanrednog širenja industrije računalnih igara, količina primjene visinskih mapa za detalje u prikazu računalnih igara daleko prestiže sve druge primjene.

4. Aplikacija Height Maker

Za praktični dio ovog završnog rada izrađena je programska aplikacija nazvana Height Maker, nazvana tako zbog funkcionalnosti učitavanja i procesiranja podataka modela u svrhu dobivanja visinske mape koja se može pohraniti u memoriju i dalje koristiti u bilo kojoj od primjena navedenih u prethodnom poglavlju.

4.1. Korištena tehnologija

Za izradu aplikacije korištena je Microsoft XNA tehnologija zbog jednostavnog načina komuniciranja sa grafičkim protočnim sustavom koristeći programsko sučelje Microsoft Direct X. Ostvarena je integracija XNA tehnologije sa Windows Forms projektom, koji je odabran zbog jednostavne implementacije korisničkog sučelja laganog za korištenje i preciznog u definiranju ponašanja. Aplikacija je pisana u programskom jeziku C#.

4.2. Logika i protočni sustav aplikacije

Microsoft.Xna.Framework biblioteka implementiraju klasu Game koja je temelj izvršavanja aplikacije. Ukratko, klasa Game je osmišljena za implementaciju računalnih igara, te se sastoji od slijedećih djelova: metode Initialize(), Load() i Unload(), koje se redom koriste za inicijalizaciju podataka, učitavanje multimedijskog sadržaja i oslobađanje memorije pri zatvaranju programa. Preostale dvije metode, Update() i Draw() se koriste za implementaciju logike i promjene podataka (Update() metoda), i iscrtavanje slikovnog sadržaja i 3D modela na ekranu (metoda Draw()). Nakon početnog i jedinog izvršavanja metoda Initialize() i Load() Po predefiniranim postavkama, interna implementacija će pokušavati pozvati obje metode 60 puta u sekundi. Ukoliko je prezahtjevno održavati taj tempo, smanjit će se broj poziva metode Draw() kako bi se metoda Update() koja sadrži bitnu logiku mogla pozivati predviđeni broj puta.

Biblioteka System.Windows.Forms sadrži klasu Form koja je koristi kao implementacija prikaza elemenata grafičkog sučelja i definicije njihovog ponašanja.

Promjenom parametara grafičkog sučelja se po potrebi pozivaju metode koje uzrokuju promjene podataka o očištu i gledištu, ili modelu.

Trenutačno podržani ulazni format modela u aplikaciji je Autodesk .fbx, koji omogućava pohranu dovoljnih i nužnih podataka za izradu visinske mape uz binarnu kompresiju sadržaja. Većina 3D programa za modeliranje pruža mogućnost spremanja scene u formatu .fbx, ukoliko ne postoji ta mogućnost, 3D model se uvijek može pohraniti u formatu Wavefront .obj, koji se potom može učitati u nekom programu koji ima podršku za pohranu .fbx formata.

Trenutačno podržani izlazni format teksture visinske mape je .png format (engl. *Portable Network Graphics*), koji omogućava pohranu podataka u kanal transparentnosti uz kompresiju bez gubitaka. Navedeni slikovni format podržan je u većini programa za uređivanje slikovnih prikaza.

4.2.1. Detalji izvedbe prikaza modela

Pri učitavanju pojedinog modela, iz podataka modela se odrede minimalne i maksimalne koordinate modela za x,y, i z os 3D prostora, koje se koriste za daljnju manipulaciju modelom i očištem.

Implementirane su dvije konceptualne vrste očišta, kojima se različito upravlja, te služe za različite svrhe. Implementacija očišta i manipulacija nad njime je spojena u klasi FreeCamera.cs, koja je ključna za jednostavnu promjenu matrice pogleda, koja se primjenjuje kao transformacija modela u koordinatni sustav očišta prije projekcijske transformacije u sustav ekrana. U daljnjem tekstu se o vrstama implementacije očišta piše kao o vrstama kamere kroz koje se vidi svijet scene.

Prva vrsta implementacije kamere je kamera s fiksnim pogledom, te se sastoji od položaja prostorne točke gledišta, matrice prostorne rotacije oko gledišta i udaljenosti od gledišta. Ova vrsta kamere služi jednostavno pregledavanje objekta koji se nalazi u gledištu iz svih kutova.

Druga vrsta implementirane kamere je slobodna kamera koja se najčešće koristi u računalnim igrama za pogled iz prvog lica. Sastoji se od matrice prostorne rotacije i položaja točke očišta. Ova vrsta kamere nema ograničenja u smislu da nije fokusirana na jednu točku gledišta, već se može slobodno rotirati.

Xna biblioteka sadrži prikladnu metodu za stvaranje matrice pogleda, koja sadrži tri jedinična ortonormalna vektora osi sustava kamere (ili očišta). Ulazni parametri metode Matrix.CreateLookAt(...) su položaj očišta, položaj gledišta i vektor smjera pozitivne Y osi prostora kamere ili vektor prema gore u prostoru kamere. Napomena: koordinatni sustav prostora u Xna implementaciji je desni koordinatni sustav. Slijedi prikaz koda obje vrste kamere.

Kod za matricu pogleda kamere s fiksnim pogledom:

```
viewMatrix = Matrix.CreateLookAt(cameraRotationMatrix.Backward * cameraRadius, c
ameraLookAt, cameraRotationMatrix.Up);
```

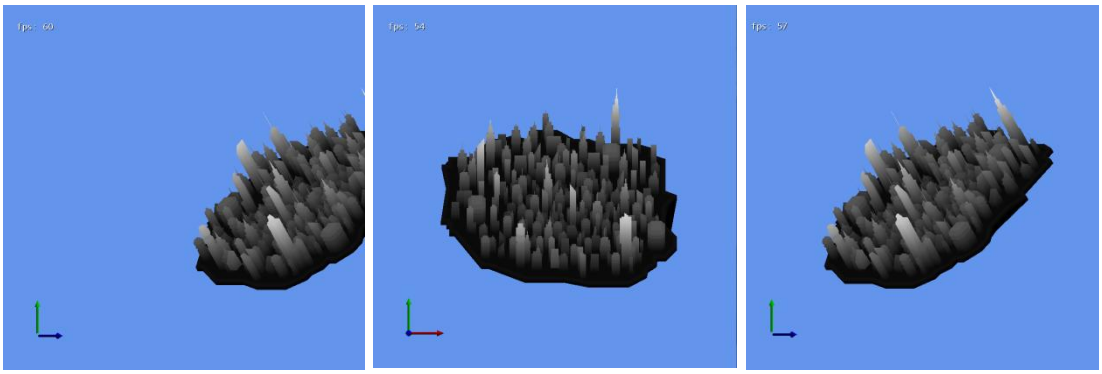
Komentar koda: cameraRotationMatrix je matrica rotacije kamere, njeno svojstvo Backward je jedinični vektor u smjeru pozitivne Z osi prostora kamere, svojstvo Up je jedinični vektor pozitivnog smjera Y osi. Varijabla cameraRadius sadrži udaljenost očišta od gledišta, a cameraLookAt sadrži točku gledišta. Budući da u ovoj vrsti kamere očište određuju udaljenost od kamere i rotacija, položaj točke očišta je umnožak unazadnog vektora rotacije i udaljenosti.

Kod za matricu pogleda slobodne kamere:

```
viewMatrix = Matrix.CreateLookAt(cameraPosition, cameraPosition + cameraRotationMatrix.
Forward, cameraRotationMatrix.Up);
```

Komentar koda: svojstvo Forward matrice rotacije sadrži vektor negativne Z osi, cameraPosition sadrži položaj točke očišta. Budući da u slobodnoj kameri ne postoji fiksno gledište, već se ono uvijek nalazi ispred promatrača, položaj točke gledišta je položaj očišta zbrojen sa vektorom smjera „naprijed“.

Slika 4.1 ilustrira razliku u primjeni slobodne i kamere s fiksnim pogledom. Srednji prikaz je izračun scene bez rotacije kamere, samo s rotacijom modela. Lijevi prikaz u pokazuje primjenu rotacije oko Y osi za 45 stupnjeva koristeći slobodnu kameru, a desni prikaz rotaciju za isti kut oko osi Y uz korištenje fiksne kamere.



Slika 4.1 – Razlike u primjeni slobodne i kamere s fiksnim pogledom

Druga matrica potrebna za transformaciju točaka modela iz 3D prostora u 2D prostor ekrana je projekcijska matrica. Za korištenje matrice projekcije odabrana je ortografska matrica zato što njena primjena na točke modela vrši njihovu paralelnu na ravninu određenu parametrima volumena pogleda. Budući da je cilj aplikacije izrada visinske mape, poželjno je da ne dolazi do perspektivnog izobličenja modela ovisno o udaljenosti od promatrača. S obzirom da se radi o paralelnom projiciranju točaka, udaljenost modela ne utječe na veličinu modela na ekranu, te su za potrebe promjene veličine modela na ekranu implementirane opcije za skaliranje modela. Kod za matricu ortografske projekcije prikladno koristi ugrađenu funkciju Xna razvojnog okruženja:

```
projectionMatrix = Matrix.CreateOrthographicOffCenter (-
graphics.GraphicsDevice.Viewport.Width / 2, graphics.GraphicsDevice.Viewport.Width / 2,gr
aphics.GraphicsDevice.Viewport.Height / 2, graphics.GraphicsDevice.Viewport.Height / 2,
0, 100000);
```

Parametri metode imaju slijedeće značenje:

```
projectionMatrix = Matrix.CreateOrthographicOffCenter(lijeva, desna, gornja, donja, prednja,
stražnja ravnina odsijecanja);
```

Naravno, zbog manjeg broja operacija na procesoru i mogućnosti slanja točaka zajedno s matricama grafičkom protočnom sustavu na izračunavanje položaja točaka na ekranu, sve transformacije nad modelom zapravo mijenjaju matricu trenutno označenog modela. Matrice transformacija modela su međusobno nepovezane, te se ne mijenjaju promjenom parametara kamere ili matrica transformacija drugih modela.

Zbog lakšeg orijentiranja modela u prostoru, implementirane su funkcije koje promjenom vrijednosti varijabli kamere i modela prilagođavaju trenutačni pogled na model. Sve metode prilagodbi su implementirane s

parametrima slobodne kamere, tako da će primjena bilo koje od njih prebaciti trenutni tip kamere u slobodnu.

Prva metoda prilagodbe zove se FitModelFromAbove(). Njena svrha je promijeniti podatke modela i kamere tako da model bude u cijelosti vidljiv odozgo. To je korisna metoda s obzirom da se visinske mape najčešće rade postavljajući očište iznad modela, gledišne izravno ispod očišta. Tako se postiže navedeni odozgo, te slijede računanje dubine točaka modela. Kod metode prilagodbe dan je u nastavku.

```
public void FitModelFromAbove()
{
  Vector3 max = CurrentModel.ExtremeMax;
  Vector3 min = CurrentModel.ExtremeMin;
  Vector3 diff = max - min;
  float biggestDistance = (diff.X > diff.Y)                    ? (diff.X > diff.Z) ? diff.X : di
ff.Z                    : (diff.Y > diff.Z) ? diff.Y : diff.Z;

  Vector2 bbsize = new Vector2(_graphicsDeviceManager.PreferredBackBufferWidth, _grap
hicsDeviceManager.PreferredBackBufferHeight);
  float smallerScreenAxis = (bbsize.X < bbsize.Y) ? bbsize.X : bbsize.Y;
  float UpPadding = 20.0f;
  float scalePadding = 0.01f;
  float uniscale = smallerScreenAxis/biggestDistance;

  UNISCALE *= (1 – scalePadding);
  CurrentModel.scale = new Vector3(uniscale);

  Camera.cameraRotation = new Vector3(MathHelper.ToRadians(-90),0,0);
  Camera.cameraPosition = new Vector3(0, max.Y * uniscale + UpPadding,0);   Camera.IsA
rcBallCamera = false;

  float newXposition =  -(min.X + max.X) / 2.0f * uniscale;
  float newZposition =  -(min.Z + max.Z) / 2.0f * uniscale;
  CurrentModel.position = new Vector3(newXposition,0,newZposition);        CurrentModel.ro
tation = Vector3.Zero;
}
```

Objašnjenje koda: varijabla *uniscale* predstavlja faktor za uniformno skaliranje modela koji se određuje na temelju omjera najveće udaljenosti između točaka modela na nekoj osi (biggestDistance) i veličine manje osi ekrana (smallerScreenAxis). Dani omjer (ili faktor) će skalirati točke modela tako da stanu u volumen kocke čija stranica iznosi duljinu manje osi ekrana, a time će i model stati na ekran. Budući da model ne mora nužno sadržavati točke na sredini 3D prostora modela, određuje se translacija modela po x i z osi koja će pomaknuti model tako da rubovi budu unutar ekrana. Rotacija modela se postavlja na nulu iz razloga što je cilj metode pogled na model originalne rotacije. Konačno, kamera se postavlja na slobodnu vrstu kamere, rotira se tako da gleda u smjeru negativne Y osi (prema dolje) i translatira se da položaj očišta bude malo iznad najviše točke modela (s obzirom na Y os) nakon uniformnog skaliranja faktorom uniscale, tako da je osigurano da se vidi cijeli model. Budući da je matrica projekcije ortografska, sve točke modela se paralelno preslikavaju i kamera gleda odozgo na model koji je cijeli prikazan na ekranu.

Druga metoda prilagodbe koristi se za namještanje položaja kamere tako da svaka točka modela bude ispred kamere, kako ne bi došlo do odsijecanja poligona koji su dijelom iza očišta. Naziv metode odgovara funkciji: FitModelFront(), iako se radi o pomicanju kamere, a ne modela, svrha funkcije je svakako postavljanje modela ispred kamere. U nastavku je dan kod ove metode.

```
public void FitModelFront()
{
   Vector3 ABC = Camera.cameraRotationMatrix.Forward;
   Vector3 camPos = Camera.cameraPosition;
   Vector4 plane = new Vector4(ABC,-Vector3.Dot(ABC,camPos));
   Vector3 scale = new Vector3(CurrentModel.worldMatrix.M11, CurrentModel.worldMatrix.M
22, CurrentModel.worldMatrix.M33);
```

```
  Vector3 max = CurrentModel.ExtremeMax;
  Vector3 min = CurrentModel.ExtremeMin;
  Vector3 diff = (max – min) * scale;

  float maxDist = (diff.X > diff.Y) ? (diff.X > diff.Z) ? diff.X : diff.Z            : (diff.Y > diff.Z) ? di
ff.Y : diff.Z;

  Vector3 vtx = (Matrix.CreateTranslation(CurrentModel.vertices[0]) *        CurrentModel.wo
rldMatrix).Translation;

  float vtx_X_plane = Vector3.Dot(vtx,ABC) + plane.W;
  if (vtx_X_plane < 0)
  {
        Camera.cameraPosition += (vtx_X_plane – maxDist) *
        Camera.cameraRotationMatrix.Forward;
  }
  else if (vtx_X_plane < maxDist)
  {
   Camera.cameraPosition-= MaxDist * Camera.cameraRotationMatrix.Forward;
  }
  else
  {
        Camera.cameraPosition += (vtx_X_plane - maxDist) *
        Camera.cameraRotationMatrix.Forward;
  }
  Camera.IsArcBallCamera = false;
}
```

Objašnjenje koda: kod računa udaljenost položaja prve točke modela u svijetu scene od ravnine koja sadrži točku očišta kamere i okomita je na smjer gledanja kamere. Rezultati se svode na dva slučaja u kojima se primjenjuje aproksimativna korekcija. U prvom slučaju, točka modela nalazi se iza bliže ravnine kamere, te u najgorem slučaju, to je rubna točka modela (sadrži neku od koordinata ekstrema modela u vektorima CurrentModel.ExtremeMax i CurrentModel.ExtremeMin) i model je rotiran tako da je ta točka najdalje od kamere u odnosu na sve druge točke modela, stoga se kamera pomiče za zbroj udaljenosti te prve točke do kamere

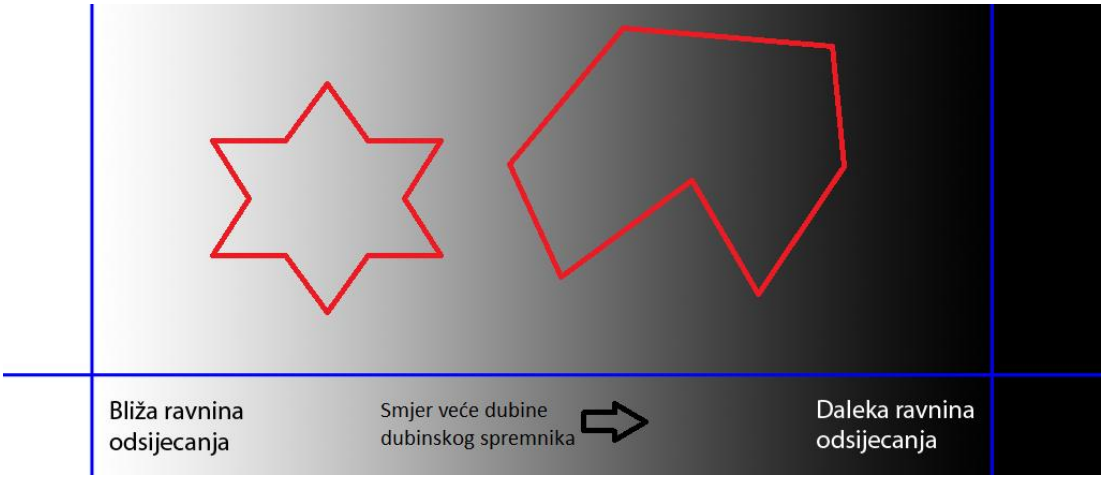
(vtx_X_plane) i udaljenosti najveće razlike osi skalirane za trenutnu vrijednost transformacijske matrice modela (maxDist). U drugom slučaju, prva točka modela je ispred kamere, no u najgorem slučaju to je također točka ekstrema, te model može biti rotiran tako da je drugi ekstrem na najvećoj mogućoj udaljenosti osi iza položaja očišta kamere, zbog čega se kamera pomiče samo za (maxDist). Ukoliko je točka modela udaljena više od maxDist s prednje strane kamere, niti jedna točka modela nije iza ravnine kamere, no neke točke mogu biti iza daleke ravnine odsijecanja, pa se kamera pomiče prema naprijed za minimalni dopušteni iznos, a da se niti jedna točka nakon pomicanja ne nađe iza kamere. Za najgori slučaj, taj iznos je jednak razlici udaljenosti prve točke modela od kamere i najveće udaljenosti modela u osima.

Postoji i treća metoda prilagodbe pogleda, nazvana FitModelAnyAngle(), koja funkcionira slično kao metoda se FitModelFromAbove(), no razlika je u tome što ova metoda ne mijenja rotaciju modela, samo pomiče model i kameru, kako bi pogled na model ostao isti, a da model bude vidljiv na ekranu.

4.2.2. Računanje visinske mape modela

Jedna od navedenih metoda izrade visinskih mapa zvana dubinsko preslikavanje bila je izvor ideje za izradu ove aplikacije. Metoda dubinskog preslikavanja je brza i efikasna, no ona nije korištena u aplikaciji zbog slijedećeg razloga: zahtjeva dodatnu čestu promjenu matrice projekcije i novo izračunavanje bliže i udaljene ravnine odsijecanja volumena ortografske matrice kako bi slikovni elementi prikaza modela obuhvaćali cijeli raspon boja od bijele (odmah ispred promatrača) do crne (udaljenost daleke ravnine odsijecanja).

Slika 4.2 ilustrira problem u postavljanju bliske i daleke ravnine u metodi dubinskog preslikavanja. Na slici je nacrtan crveni rub modela, a unutrašnjost okvira je prozirna kako bi se vidjela pretpostavljena boja točaka modela na toj dubini dubinskog spremnika. Ukoliko želimo da svaki model u trenutku prikazivanja na točkama koje su ekstremi udaljenosti na z osi nakon projekcije u sustav ekrana ima krajnje boje (crnu i bijelu), vidljivo je kako bi se trebale zadati nove vrijednosti ravnina u matrici projekcije.



Slika 4.2 – Prikaz boja točaka modela s obzirom na dubinu nakon projekcije u sustav ekrana

Drugi razlog koji je pridonio odluci na primjenu drukčije metode je želja autora da se model može prikazati iz drugih kutova, nakon što su boje točaka modela određene ovisno o željenom kutu.

Odabrana metoda funkcionira na slijedeći način: svakoj točci modela pridruži se podatak o boji, s obzirom na položaj očišta i transformiranog

položaja te točke, odnosno njene udaljenosti od ravnine koja sadrži očište i ima normalu jednaku smjeru pogleda. Točkama najbližim ravnini očišta pridjeljuje se bijela boja, te što je točka više udaljena, dodjeljuje joj se tamnija boja.

U grafičkom protočnom sustavu će se izračunati položaj svake točke modela u prostoru ekrana, te će se za svaki slikovni element ekrana interpolacijom izračunati boja neke točke poligona koja je vidljiva na ekranu. Prethodna rečenica ukazuje na automatsku dodjelu ispravne boje s obzirom na položaj očišta i točaka modela te boje pojedinih točaka.

Implementirane su dvije metode koje omogućavaju promjenu boja točaka modela. Prva metoda zove se RecolorFromY() te se koristi kako bi se boje modela postavile s obzirom na originalne koordinate modela, i to na način da točka s najvišim iznosom y koordinate bude bijela, a ona s najnižim crna. Slijedi prikaz koda.

```
public void RecolorFromY(List<Vector3> vertices, out List<Color> colors)
    {
        colors = new List<Color>();
        Vector3 v;
        float raspon = ExtremeMax.Y-ExtremeMin.Y;
        for (int i = 0; i < vertices.Count; i++)
        {
            float boja = (float)(vertices[i].Y - ExtremeMin.Y          ) / raspon;
            v = new Vector3(boja);
            colors.Add(new Color(v));
        }
    }
```

Objašnjenje koda: prije prolaza kroz sve točke modela odredi se raspon u koordinatama y osi koji obuhvaća točke modela. Za to se koriste podaci modela ExtremeMax i ExtremeMin koji sadrže najveće i najmanje koordinate točaka modela te se određuju jednom, pri učitavanju modela.

Slijedi određivanje omjera udaljenosti točke od minimalne y koordinate i ukupnog raspona, što daje broj između 0 i 1. Bijelu boju moguće je zapisati koristeći vektor (1.0,1.0,1.0) ako se koristi decimalni zapis i raspon od 0 do 1

ili kao vektor (255,255,255) ako se koriste cijeli brojevi i raspon od 0 do 255. Crna boja uvijek se zapisuje kao vektor (0,0,0). Vektor sa svim komponentama jednakom prethodno određenom omjeru se postavlja kao boja trenutne točke. Ponavljajući postupak za svaku točku modela postiže se obojanost gradijentom bijele i crne boje po y osi, što predstavlja intenzitet visine na dubinskoj mapi koja se može izraditi od takvog prikaza modela.

Druga metoda koja mijenja podatke o boji točaka je metoda ChangeHeightColoringToCameraAngle(). Ona se koristi kako bi se točkama pridijelila nova boja ovisno o udaljenosti od očišta, no neovisno o kutu modela ili kamere. Ova metoda omogućava proizvoljno pozicioniranje kamere i modela te bojanje točaka tako da konačni prikaz izgleda kao visinska mapa modela iz trenutnog kuta pogleda. Metoda funkcionira na sličnom principu kao i RecolorFromY(). U nastavku slijede dijelovi koda i objašnjenja.

Prvi dio koda traži ekstreme u z koordinatama koje točke modela zauzimaju u svijetu kamere. Petlja svaku točku modela prvo transformira u prostor scene:

```
Vector3 vertex = CurrentModel.vertices[currentIndex];
Vector3 vtx =(Matrix.CreateTranslation(vertex)* CurrentModel.worldMatrix);
```

Zatim slijedi transformacija u prostor kamere:

```
Matrix camRotInvert = Matrix.Invert(Camera.cameraRotationMatrix);
        Matrix camRotInvert = Matrix.Invert(camRot);
vtx =(Matrix.CreateTranslation(vtx) * camRotInvert).Translation;
```

Budući da nam treba samo Z koordinata točke transformirane u prostor kamere:

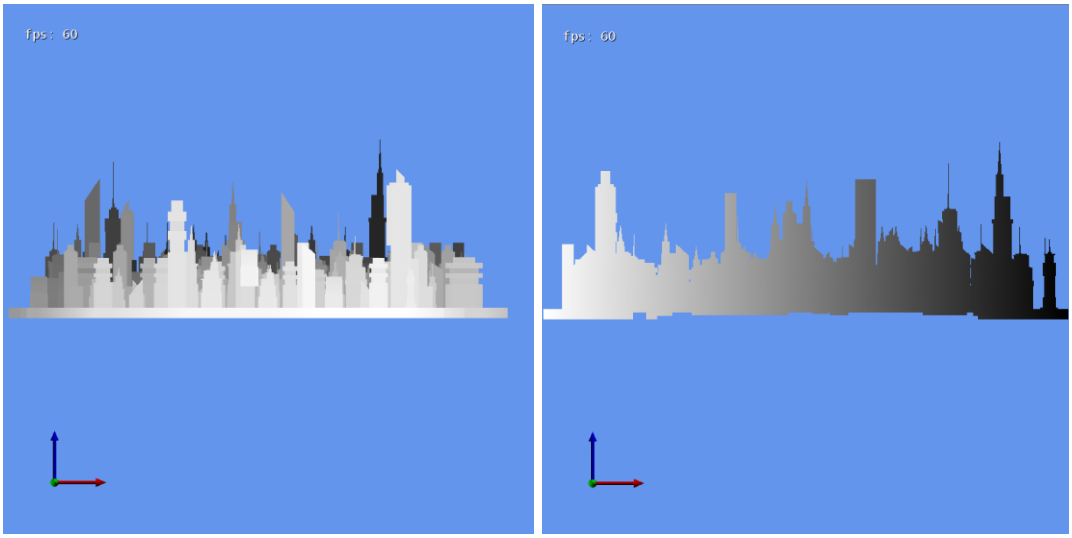
```
float vtxZ = vtx.Z;
```

Slijedi usporedba z koordinate trenutne točke sa trenutno najvećom i najmanjom zapamćenom z koordinatom u svrhu traženja ekstrema i

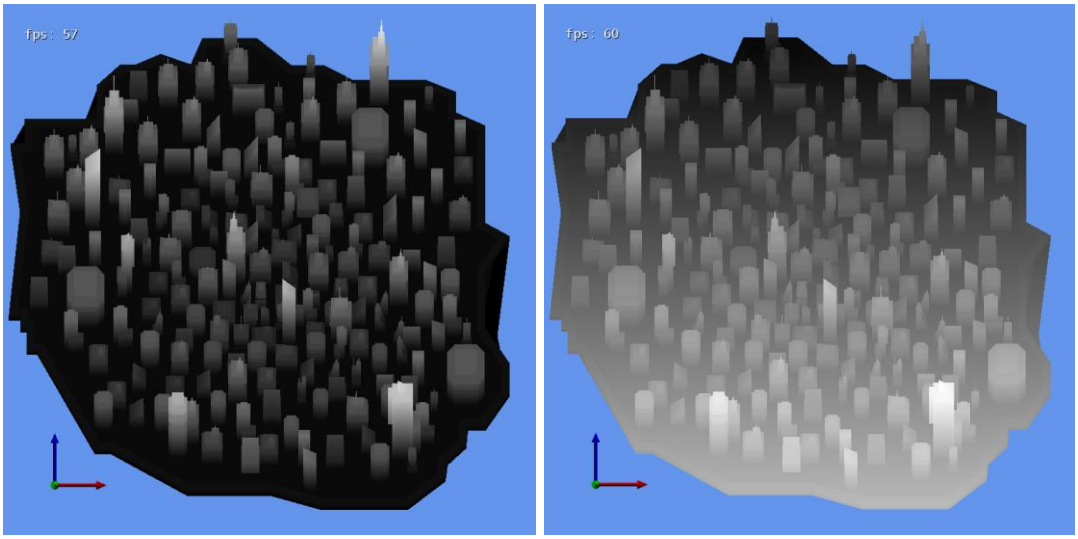
određivanja raspona isto kao u metodi RecolorFromY(). Ostatak koda ove metode je identičan kao u RecolorFromY() metodi, s iznimkom jednog reda određivanja boje trenutne točke. U ovoj metodi boja trenutne točke se temelji na rasponu i udaljenosti koordinate z točke transformirane u prostor kamere od rubova raspona. Koriste se pohranjeni podaci prethodnih tranformacija vtxZ za svaku točku.

```
float boja = (vtxZ[currentIndex] - minAxy.Z ) / raspon;
```

Slika 4.3 prikazuje primjenu ove metode na prednji kut pogleda na model grada. Na lijevom prikazu vidi se primjena metode na model iz pogleda na model s prednje strane. Desni prikaz je model nepromijenjenih podataka o točkama u odnosu na prethodnu sliku, to s rotacijom modela za 90 stupnjeva oko Y osi. Na tom prikazu je izvrsna vidljivost ispravnog pridruživanja boja u potpunom rasponu od bijele do crne boje, s obzirom na kut pogleda, a vidi se i interpolacija boja poligona uzduž bivše z osi pogleda, točnije: ne vide se rubovi poligona jer su točke bile na istoj dubini te im je dodijeljena ista boja.



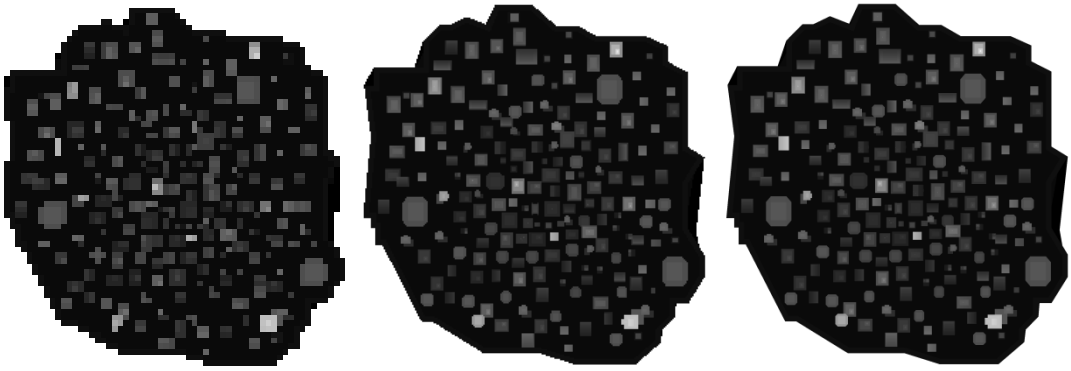
Slka 4.3 – Prikaz rezultata primjene metode ChangeHeightColoringToCameraAngle()



Slika 4.4 - Prikaz razlike u primjeni metoda *RecolorFromY()* (desni prikaz) i *ChangeHeightColoringToCameraAngle()* (desni prikaz)

Aplikacija može u bilo kojem trenutku pohraniti prikaz scene iz trenutnog pogleda na model u proizvoljnoj rezoluciji ograničenoj u obje dimenzije rasponom između 16 i 4096 slikovnih elemenata. Budući da se boja pojedinog slikovnog elementa koji pripada modelu računa interpolacijom boja u točkama pripadajućeg poligona, povećanjem izlazne rezolucije teksture ne gubi se na detaljima, samo se dobiva blaži prijelaz boja. Nakon pritiska gumba na korisničkom sučelju koji se koristi za spremanje prikaza (Save Screen), stvara se tekstura čije dimenzija odgovara postavljenoj izlaznoj rezoluciji. Ta tekstura se šalje u grafičkom protočnom sustavu te se koristi kao memorijsko odredište za zapis scene umjesto izlaznog toka podataka o slikovnim elementima.

Slika 4.5 prikazuje izlazne teksture različite rezolucije skalirane na jednaku veličinu u ovom dokumentu (otprilike 180 slikovnih elemenata), koje predstavljaju pogled odozgo na prethodno korišteni model grada. Redoslijed rezolucija slika je 64, 256 i 2048 slikovnih elemenata po dimenziji visine i širine slike.



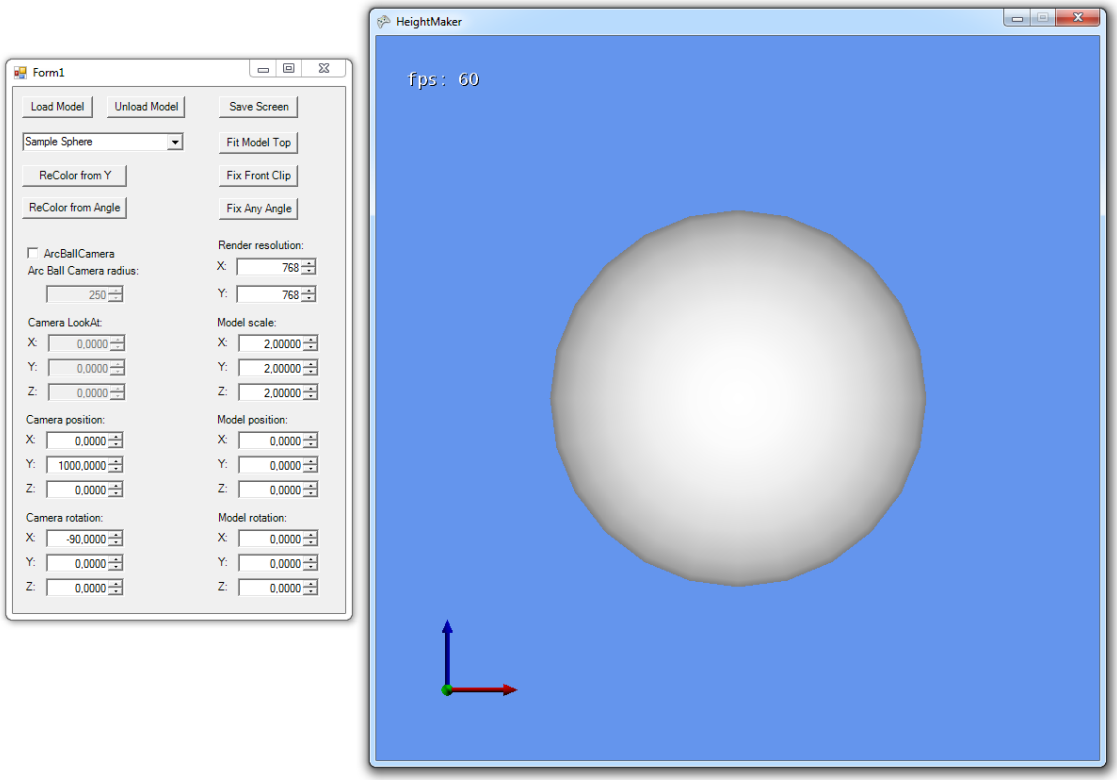
Slika 4.5 – Prikaz rezultantnih visinskih mapa različitih rezolucija dobivenih aplikacijom Height Maker

4.3. Upute za korištenje aplikacije

Aplikacija omogućava učitavanje i uklanjanje modela iz memorije tijekom izvođenja, osnovnu manipulaciju pogleda u 3D prostoru, translaciju, rotaciju i skaliranje modela, postavljanje boja svim točkama modela ovisno o udaljenosti od očišta i odnosu položaja očišta i modela. U bilo kojem trenutku izvođenja moguće je spremiti trenutačni prikaz modela u teksturu koja se pohranjuje u memoriju.

Pri pokretanju aplikacije otvaraju se dva prozora, jedan sadrži korisničko sučelje aplikacije, drugi se koristi za iscrtavanje modela u 3D prostoru. Po predefiniranim postavkama, u aplikaciju se učitavaju dva modela koji su pohranjeni zajedno s podacima aplikacije: model kocke (od 6 točaka) te aproksimacija modela sfere (266 točaka). Nad točkama oba modela je primjenjena funkcija postavljanja boja uzduž Y osi, a kamera je postavljena da gleda prema negativnoj Y osi, tako da će se vidjeti svijetlija, viša strana modela.

Odmah nakon pokretanja aplikacije moguće je spremiti visinsku mapu postojeći modela, ili mijenjati sve ponuđene parametre. Slika 4.6 prikazuje izgled oba ekrana nakon pokretanja programa i prebacivanja trenutnog modela na model kugle.



Slika 4.6 – Prikaz izgleda prozora aplikacije Height Maker

Pritiskom na gumb označen tekstom „Load Model“ otvara se prozor koji prikazuje direktorije sadržaja računala, te prikazuje prisutne .fbx datoteke. Izborom neku .fbx datoteke i pritiskom na gumb „Open“ otvara se model nad kojim se primjenjuju funkcije pogleda odozgo i bojanja točaka uzduž y osi. Pritiskom na gumb „Unload model“ iz memorije se briše trenutno označeni model. Odabirom željene rezolucije prikaza i pritiskom na „Save Screen“ otvara se prozor koji prikazuje direktorije sadržaja računala. Nakon odabira željenog direktorija, imenovanja nove slikovne datoteke i pritiskom na „Save“, trenutni prikaz modela se pohranjuje u teksturu u radnoj memoriji koja se sprema pod odabranim imenom na odabranoj lokaciji u trajnu memoriju.

5. Mape normala

Jedna od najpopularnijih tekstura posebne primjene za dodavanje detalja konačnom prikazu scene je mapa normala. To je dvodimenzionalna tekstura u kojoj su pomoću intenziteta boja pohranjeni podaci koji se interpretiraju kao usmjereni vektori 3D prostora. U slijedećim poglavljima bit će opisane vrste, načini dobivanja i primjene mape normala.

Samo za usporedbu, osnovna razlika mape normala i visinske mape u primjeni dodavanja detalja (neravnina) površini je što se primjenom visinske mape originalni vektor normale nekog slikovnog elementa modificira s obzirom na izvornu vrijednost, a primjenom mape normala izvorni vektor se potpuno zamjenjuje vektorom određenim iz slikovnog elementa mape normala.

Druga, vidljivija razlika je u izgledu. Dok su visinske mape crno-bijele teksture, mapa normala zauzima sva tri kanala boje i često sadrži različite vektore, pa izgleda puno zanimljivije, ali ju je puno teže interpretirati iz samog pogleda na teksturu. Svaki slikovni element visinske mape sadrži jedan od 256 različitih intenziteta boje (u slučaju visinske mape koja koristi samo jedan kanal), a svaki slikovni element mape normala ima pohranjen jedan od 16.777.216 vektora. Naravno, moguće je iskoristiti i 4. kanal teksture, transparentnost, za dodatnu kvantizaciju vektora normala, no to se u praksi ne radi, već se u kanal transparentnosti često pohranjuje visinska mapa, kako bi se mogla zajedno s mapom normala iskoristiti u popularnoj primjeni, metodi preslikavanja zaklanjanja uslijed paralakse, koja je opisana u poglavlju „Primjene visinskih mapa“.

5.1. Značenje boja mape normala

S obzirom da se slikovne elemente mape normala interpretira kao vektore, potrebno je definirati značenje pojedine komponente boje. Komponenta crvenog kanala predstavlja X os, komponenta zelenog Y, a plavog Z os. Nakon definiranja značenja boja, potrebno je definirati raspon vektora i neutralni vektor. Budući da mapa normala služi za pohranu vektora

normala, svi predstavljeni vektori moraju biti normalizirani, stoga se raspon interpretacije ograničava na [-1,+1] po svakoj komponenti boje. U nastavku ovog poglavlja će biti opisane dvije vrste mapa normala, a to su mape normala prostora scene i tangencijalnog prostora. Ukratko, mape normala prostora mogu sadržavati vektore bilo kojeg smjera, dok mape tangencijalnog prostora sadrže samo vektore s pozitivnom komponentom Z osi (plave boje). Raspon interpretacije obje vrste mapa normala je isti, te je za neutralan slikovni element koji predstavlja vektor na sredini raspona interpretacije, v(0,0,0), odabrana boja na sredini raspona intenziteta boja (koji iznosi [0,255]), a to je boja (127.5,127.5,127.5), no zbog kvantizacije zapisa intenziteta pomoću 8 bita, boja je zaokružena na (128,128,128) ili konkretno srednje sivu boju.

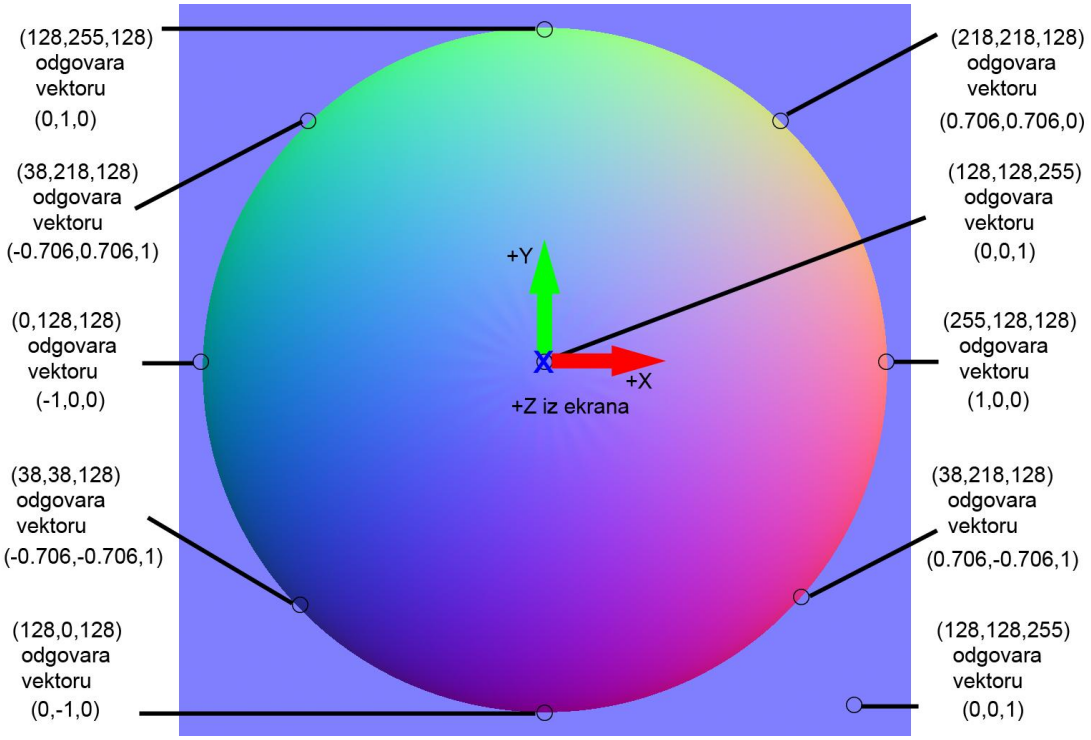
S obzirom na neutralni vektor i raspon vektora, intenzitet boje slikovnog elementa bilo koje mape normala se pretvara u odgovarajući vektor normale na slijedeći način:

Vektor normale = 2 * (boja slikovnog elementa – neutralna boja) / raspon zapisa boja

Napomena: ovisno o korištenoj biblioteci za grafičke funkcije raspon boja može biti [0,255] ili [0,1], a ponekad ovisi o tipu varijable korištene u tim funkcijama. Formula za pretvorbu vektora normale u boju se trivijalno izvodi iz gornje formule:
Boja slikovnog elementa = 255 * (vektor normale / 2) + neutralna boja

Slika 5.1 prikazuje tangencijalnu mapu normala izrađenu praćenjem zrake iz modela ravnine na model polukugle (koji je postavljen iznad modela ravnine u sceni). Potrebno je napomenuti da je neutralna boja za mape normala tangencijalnog prostora upravo nepromijenjeni vektor normale na tangencijalnu površinu, koji iznosi (0,0,1), ili preslikano u boju slikovnog elementa po gornjoj formuli: boja = 255 * ((0,0,1) / 2) + (128,128,128) = (128,128,255). Većina mapa normala ove vrste će mjestimice imati slikovne

elemente vrijednosti (**127**,**127**,255), što će izazvati probleme u nekim primjenama takve teksture, potrebno je biti svjestan različitih implementacija zaokruživanja rezultata dijeljenja 255 sa 2 i posljedica takve pohrane vektora.



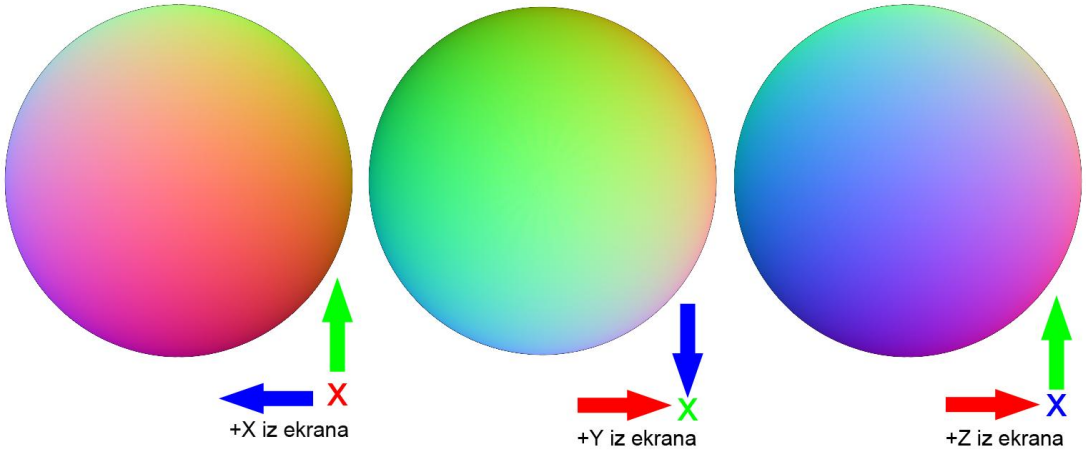
Slika 5.1 – Prikaz spektra mape normala tangencijalnog prostora

Na slici 5.1 prikazane su aproksimacije vektora na tri decimale dobivenih izračunom po gornjoj formuli iz intenziteta slikovnog elementa. Potrebno je napomenuti kako je slika samo ilustracija spektra tangencijalnih mapa normala. Kako bi se lakše vizualiziralo značenje pojedinih boja i dobivenih vektora, nacrtane su osi tangencijalnog prostora.

Kao što se može primjetiti, niti jedan slikovni element nema komponentu plave boje manju od 128 (u teoretski savršenoj mapi normala tangencijalnog prostora), jer je svrha mape normala povećati količinu vidljivih detalja na poligonu, te za pravilni izračun intenziteta osvjetljenja, tangencijalna mapa normala mora imati vektor pozitivan u z osi. Četiri vektora koja imaju z komponentu jednaku nuli su takva zbog aproksimacije idealnosti, prave vrijednosti slikovnih elemenata blago odudaraju od navedenih, no ovo je samo ilustracija. Komponente vektora koje iznose (0.706) ili (-0.706) su prikaz problematičnosti kvantizacije i ograničenja mapa

normala, s obzirom da se u intenzitete boja trebala pohraniti komponenta apsolutnog iznosa $\left(\frac{1}{\sqrt{2}}\right)$.

Slika 5.2 prikazuje spektre boja mapa normala prostora scene. Slika se sastoji od tri prikaza, od kojih svaki odgovara mapi normala polukugle gledane iz odgovarajuće osi. Na slici su dodatno označeni smjerovi osi scene za svaki prikaz. Boje svakog spektra se mijenjaju prema prethodno opisanom principu sa slike 5.1. Potrebno je primjetiti kako spektar mape normala prostora scene gledana iz pozitivne z osi izgleda poput spektra mape normala tangencijalnog prostora upravo iz razloga što u tangencijalnom prostoru smjer od površine prema tangencijalnom „gore“ je pozitivna z os.



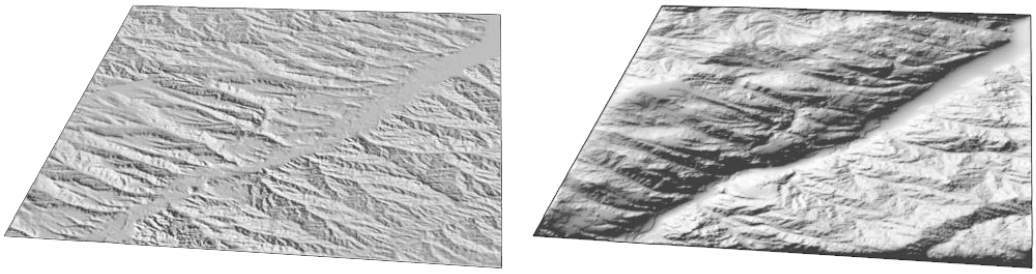
Slika 5.2 – Prikaz spektara boja mapa normala prostora scene, gledano iz pozitivnog smjera osi X, Y i Z

5.2. Kratka usporedba s visinskim mapama u primjeni tehnike preslikavanja neravnina

Na početku ovog poglavlja spomenuta je osnovna razlika u primjeni visinskih mapa i mapa normala na preslikavanje neravnina površine: primjena visinske mape alternira normalu poligona zbrajanjem s normalom određenom visinskom mapom, dok u primjeni mape normala izračunati vektori u potpunosti zamjenjuju normale poligona. U nastavku su prikazani

rezultati primjene isključivo mape normala ili visinske mape na ravninu preslikavanjem neravnina, kako bi se naglasile razlike u kvaliteti primjene.

Lijevi dio slike 5.3 prikazuje primjenu visinske mape sa slike 5.11 na model kvadrata tehnikom preslikavanja neravnina. Desni dio slike prikazuje primjenu mape normala izrađenu od iste visinske mape na isti model kvadrata. Intenzitet i kut osvjetljenja je isti na oba prikaza.



Slika 5.3 – Prikaz razlike u primjeni tehnike preslikavanja neravnina korištenjem visinske mape (lijevo) i mape normala (desno)

Sa slike 5.3 jasno je vidljivo koliko mapa normala ima jači (potpuni) utjecaj na normale poligona na koje je preslikana, te koliko to pridonosi stvaranju dojma o većoj količini detalja na prikazu scene. Visinska mapa je alternirala normalu poligona (koja je u ovom slučaju iznosila (0,0,1) u tangencijalnom prostoru poligona, a (0,1,0) u prostoru scene) prema ranije navedenoj formuli:

KonačnaNormala = Normaliziraj(UlaznaNormala + NormalaIzVisinskeMape);

Pri izračunavanju konačne normale očito je da normala poligona (Ulazna normala) ima jak utjecaj na rezultat, te je ključni razlog zašto je komponenta vektora usmjerena prema gore ostala predominantna na cijelom prikazu nakon primjene visinske mape. U prikazu primjene mape normala, vidljivo je

da su normale svih slikovnih elemenata nepovezane s normalom poligona, već potpuno preuzete od vektora pohranjenih u mapi normala.

Prikazana razlika kvalitete između visinskih mapa i mapa normala u primjeni preslikavanjem površinskih neravnina razlog je brzom prihvaćanju mapa normala kao boljim načinom za primjenu u industriji računalnih igara po cijelom svijetu.

5.3. Vrste mapa normala

Postoje dvije osnovne vrste mapa normala koje se koriste za preslikavanje neravnina. To su mapa normala prostora scene i mapa normala tangencijalnog prostora.

5.3.1. Mapa normala prostora scene

Prva vrsta mapa normala često se naziva i mapa prostora modela ili lokalnog prostora. Mapa se koristi za zapisivanje vektora normala u prostoru objekta, što znači da se u njoj mogu pojaviti sve kombinacije vrijednosti kanala boje, za svih 16.777.216 vektora. Problem u primjeni ovih mapa je da su specifične za objekt u određenom položaju u smislu orijentacije. To znači da će rezultat primjene ovih mapa izgledati ispravno s obzirom na izvore svjetlosti u sceni samo ako je model netransformiran, skaliran ili translatiran. Transformacije rotacije i deformacije mijenjaju orijentaciju modela, te bi primjena mapa normala lokalnog prostora na tako transformirane modele bila puna netočnih rezultata računanja intenziteta osvjetljenja. Ova vrsta mapa je jedinstvena za svaki objekt, u smislu da će različiti oblici davati krive rezultate za istu mapu normala lokalnog prostora, za razliku od druge vrste mapa normala, koja je primjenjiva na sve površine s jednako kvalitetnim i točnim prikazom detalja. Dodatno, ovu vrstu mapa teže je uređivati u programima za obradu slika zbog blage promjene boja kroz teksturu, te je teško postići ispravne rezultate. Zbog navedenih ograničenja, mape normala lokalnog prostora se vrlo rijetko koriste u praksi. Ova vrsta mape normala je

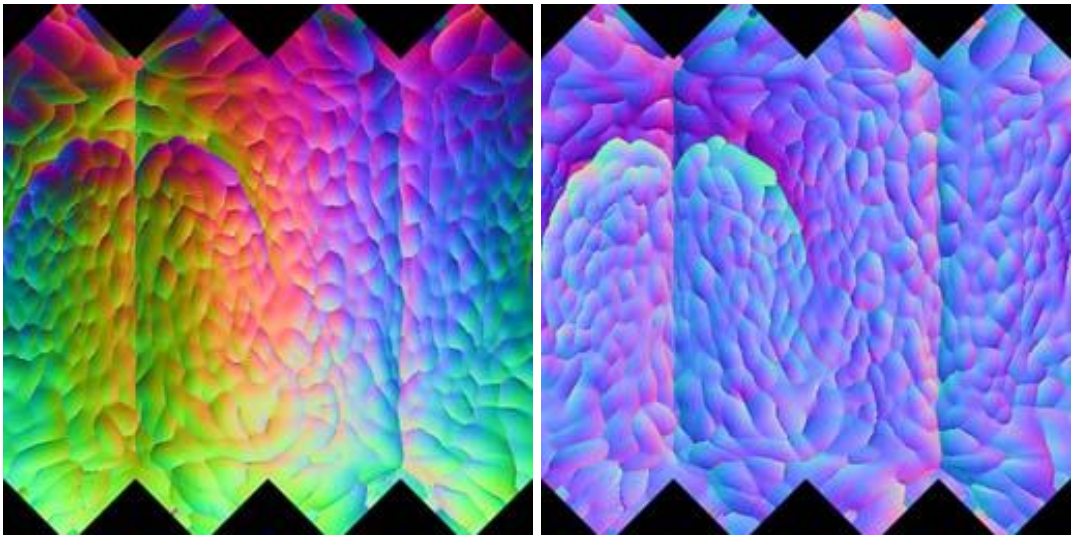
prepoznatljiva po rasponu duginih boja koje se pojavljuju na njoj u nekom pravilnom rasporedu, ovisno o tipu površine čije normale su preslikane.

5.3.2. Mapa normala tangencijalnog prostora

Ova vrsta mapa normala je dominantna u korištenju za tehniku preslikavanja neravnina mapama normala. Naziv tangencijalni prostor odnosi se na lokalni prostor svakog djela poligona koji u grafičkom protočnom sustavu bude određen kao slikovni element za prikaz na ekran. Tangencijalni prostor određen je jediničnim ortonormalnim vektorima koji se nazivaju normala, tangenta i bitangenta (ponekad se koristi naziv binormala). Normala je vektor okomit na površinu, tangenta je vektor okomit na normalu i predstavlja x os u lokalnom prostoru, bitangenta je okomita na prethodna dva vektora i koristi se kao y os lokalnog prostora.

Kao što je spomenuto, mape normala tangencijalnog prostora se mogu ponovno iskorištavati na različitim objektima, te davati ispravne rezultate i na deformiranim te rotiranim objektima. To svojstvo imaju upravo zato što se u njih pohranjuju vektori koji su usmjereni isključivo iz površine, s različitim usmjerenjem, no nikad nemaju z komponentu manju od 0. Zbog pozitivne ili neutralne komponente z, svaki slikovni element mape normala tangencijalnog prostora ima intenzitet kanala plave komponente veći ili jednak 128, zbog čega su tangencijalne mape normala prepoznatljive po prevladavajuće plavoj boji. Ove mape normala je zbog istog razloga lakše uređivati i spajati u programima za manipuliranje slikovnim elementima, jer su prijelazi u bojama razumljiviji i više očiti, a i sam broj različitih boja u ovim mapama je manji, jer se nikad neće pojaviti sve boje s plavom komponentom manjom od 127, dok takve boje prosječno prekrivaju polovicu mape normala lokalnog prostora.

Na slici 5.4 prikazane su dvije navedene vrste mapa normala: mapa lokalnog prostora (lijevo) i mapa tangencijalnog prostora (desno) nastale od istog modela.



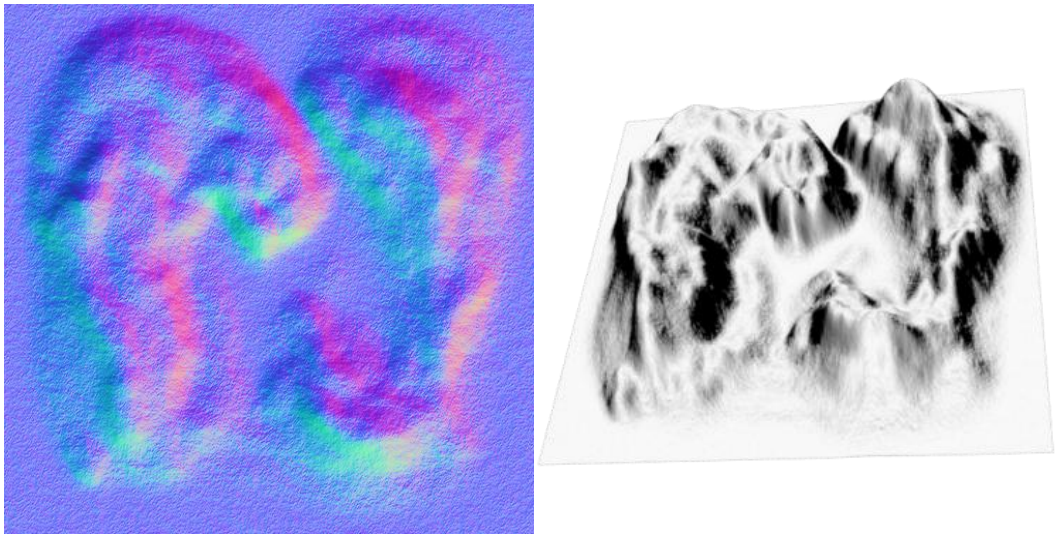
Slika 5.4 – Prikaz lokalne (lijevo) i tangencijalne mape normala (desno) [19]

5.3.3. Rezolucija i pravilnost vrijednosti mape normala

Kao i kod visinskih mapa, u primjeni mapa normala vidljiva razlika u kvaliteti prikaza često je rezultat nedovoljne rezolucije slike. Budući da mape normala zauzimaju tri puta više memorije u odnosu na prosječne visinske mape iste rezolucije, potrebno je pažljivije birati između raspoloživih rezolucija za svaku mapu normala, uzimajući u obzir kvalitetu izgleda i performanse.

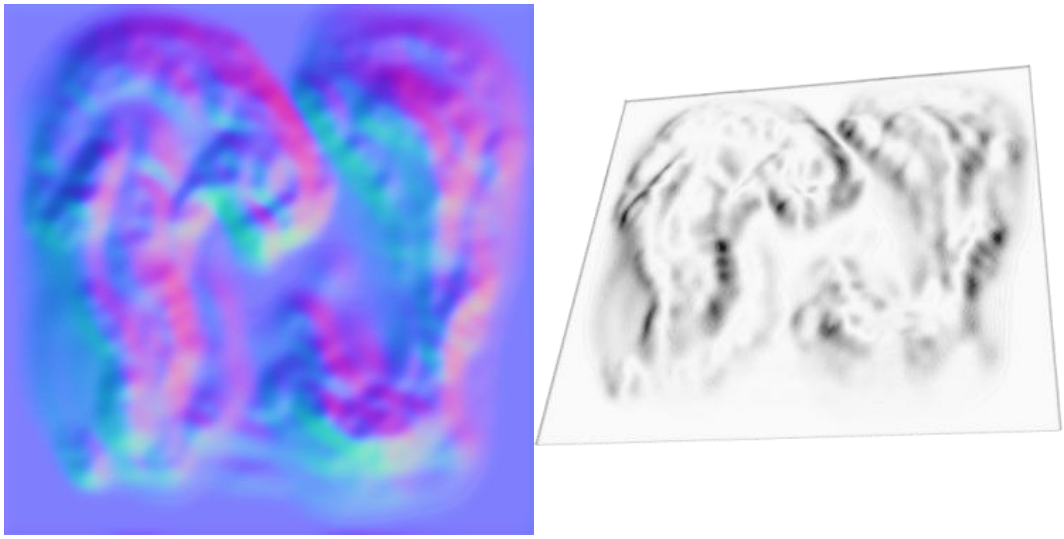
Poput primjera u uvodu teorije visinskih mapa, u nastavku su dana dva loša ekstrema u odabiru vrijednosti. Prvi primjer je mapa normala niske rezolucije i visoke frekvencije alterniranja iznosa i smjera normala interpretiranih bojama slikovnih elemenata teksture. Primjena takve mape normala rezultirat će predinamičnim i nepreciznim prikazom, kao rezultatom prečestih promjena u intenzitetu osvjetljenja modela, što je izravna posljedica šuma prisutnog u vrijednostima slikovnih elemenata mape normala. Lijevi dio slike 5.5 prikazuje primjerak mape normala s navedenim svojstvima. Desni dio iste slike prikazuje rezultat primjene teksture preslikavanjem normala na ravnu plohu uz preslikavanje premještanjem geometrije, za koje se koristila

visinska mapa prikazana u lijevom dijelu slike 5.3. Korištena mapa normala dobivena je izradom iz iste visinske mape.



Slika 5.5 - Primjer visoke frekvencije i niske rezolucije mape normala

Drugi primjer atributa nekvalitetne mape normala su visoka rezolucija i maleni razmak u vrijednostima slikovnih elemenata, koji se može dobiti smanjenjem raspona intenziteta visinske mape iz koje se dobiva mapa normala. Lijevi dio slike 5.6 prikazuje mapu normala visoke rezolucije i smanjene alternacije vrijednosti slikovnih elemenata, dobivenu iz visinske mape s lijevog dijela slike 5.2. Desni dio slike je dobiven korištenjem prethodno navedene mape normala i visinske mape sa slike 5.2. Zbog veće rezolucije, rezultat učitavanja teksture i čitanje njenih podataka u protočnom sustavu traje dulje, a manji raspon vrijednosti rezultira nedovoljnom naglašenošću razlika u nagibu površine.



Slika 5.6 – Primjer niske frekvencije i visoke rezolucije mape normala

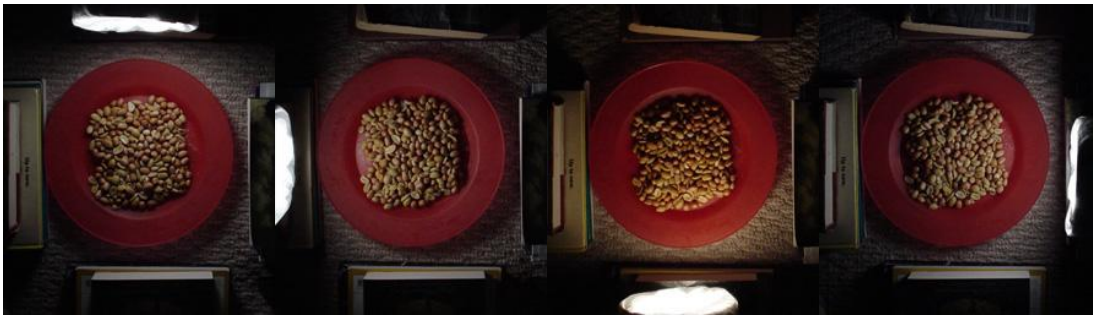
5.4. Načini izrade mapa normala

Postoje različiti načini izrade mapa normala, koji variraju u kvaliteti, zahtjevnosti resursa, vremenu potrebnom za izradu, i naravno složenosti izrade. Načini izrade koji su navedeni u nastavku uglavnom su dostupni prosječnom korisniku računala s pristupom internetu, s iznimkom prvog načina.

5.4.1. Izrada mape normala obradom fotografija

Jedan od najrjeđih i najzanimljivijih načina izrade mapa normala je način izrade u kojem se koriste zamračena soba, ručna svjetiljka, digitalna kamera i program za manipulaciju slika. Postupak je prilično jednostavan, te njegov opis slijedi u koracima, uz popratne slike, preuzete sa Interneta [20].

1. korak: potrebno je uslikati odabrani objekt odozgo 4 puta, tako da je u svakoj fotografiji osvijetljen s iste visine, istim intenzitetom, no sa četiri različite strane. Preporučena visina je ona u kojoj je dio udaljen od svjetla djelomično zasjenjen, no ne smije dolaziti do prevelikog zaklanjanja objekta vlastitom siluetom. Slika 5.7 prikazuje postavljene „scene“ objekta i izvora svjetla prije fotografiranja.

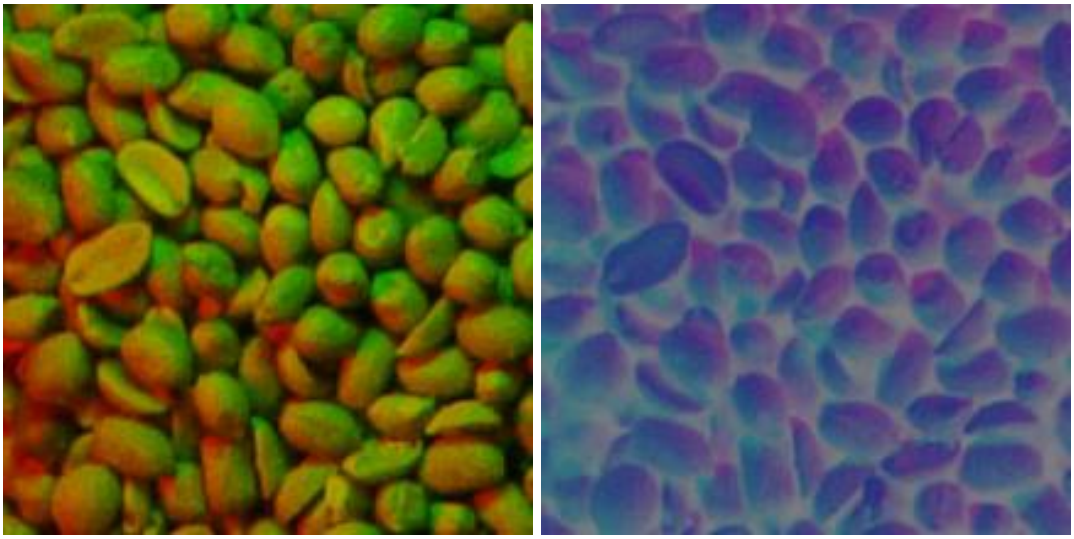


Slika 5.7 – Prikaz objekta za fotografiranje i izvora svjetla za 4 različita smjera.

2. korak: fotografije objekta se filtrom sivog spektra pretvore u crno-bijele fotografije. Slika 5.8 prikazuje dijelove fotografija koji su odabrani za izradu mape normala (isti dio fotografije objekta za svaki od 4 kuta osvjetljenja).

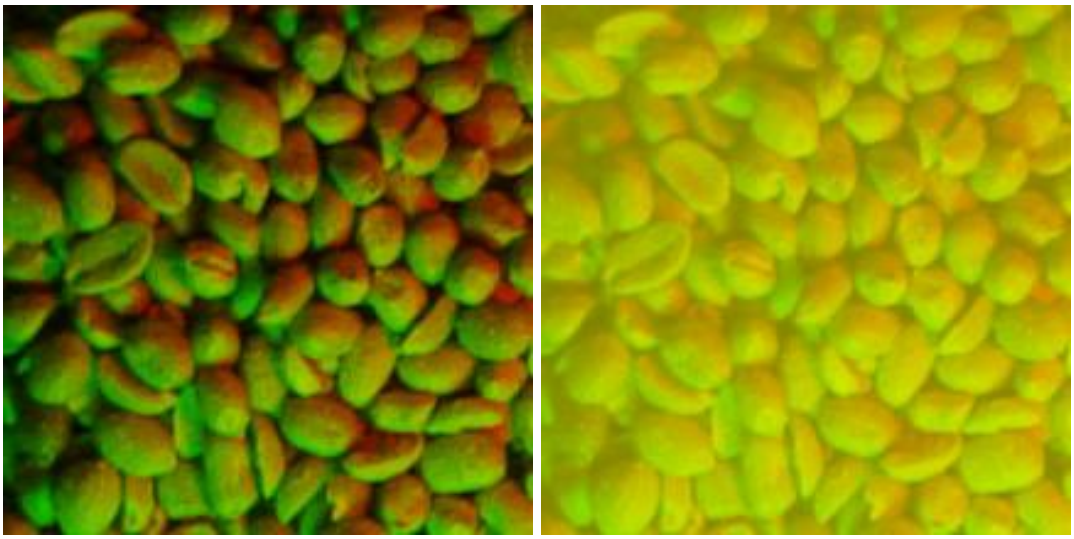
Slika 5.8 – Prikaz crno-bijelih fotografija objekta za izradu mape normala

3. korak: slika objekta osvijetljenog odozdo spoji se sa slikom objekta osvjetljenog slijeva u novu sliku na način da prva slika predstavlja komponentu zelene boje, a druga komponentu crvene boje. Nakon toga, razine intenziteta slike promijene se na način da se raspon intenziteta iz [0-255] preslika u raspon [127,0]. Jednak učinak bi imao postupak oduzimanja vrijednosti svih komponenti slikovnih elemenata od maksimalnog intenziteta (255), te dijeljenje svih vrijednosti s 2. Slika 5.9 prikazuje spoj slike osvijetljene odozdo u zelenoj komponenti i slike osvijetljene slijeva u crvenoj komponenti boje (lijevo) , te istu sliku nakon promjene intenziteta slikovnih elemenata (desno).



Slika 5.9 – Prikaz spoja slika objekata osvjetljenih slijeva i odozdo

4. korak: slično kao u koraku 3, slike objekta osvjetljenog odozgo i s desne strane se pohrane u zelenu i crvenu komponentu slike, tim redoslijedom. Zatim se raspon intenziteta preslika iz [0,255] u raspon [128,255]. Isti rezultat postiže se dijeljenjem intenziteta svih slikovnih elemenata s 2 te dodavanjem vrijednosti 128, za svaku komponentu boje. Slika 5.10 prikazuje spoj slike osvjetljene odozgo u zelenoj komponenti i slike osvijetljene s desne strane u crvenoj komponenti boje (lijevo), te istu sliku nakon promjene intenziteta slikovnih elemenata (desno).



Slika 5.10 – Prikaz spoja slika objekata osvjetljenih s desne strane i odozgo

Korak 5: nakon dobivanja dvije rezultante slike, prikazane u desnim dijelovima slika 5.9 i 5.10, preostaje operacija spajanja intenziteta njihovih slikovnih elemenata na poseban način, koji se u programu Adobe Photoshop naziva „overlay“ filtrom. U tom efektu ulazni parametri su dvije teksture, jedna

koja služi kao maska i druga, koja se maskira. U ovom slučaju, tekstura s ulogom maske je ona na desnom dijelu slike 5.10, a tekstura koja će biti maskirana je tekstura s desnog dijela slike 5.9. Formula „overlay“ filtra je dana u nastavku:

Za svaki slikovni element, za svaku komponentu boje:

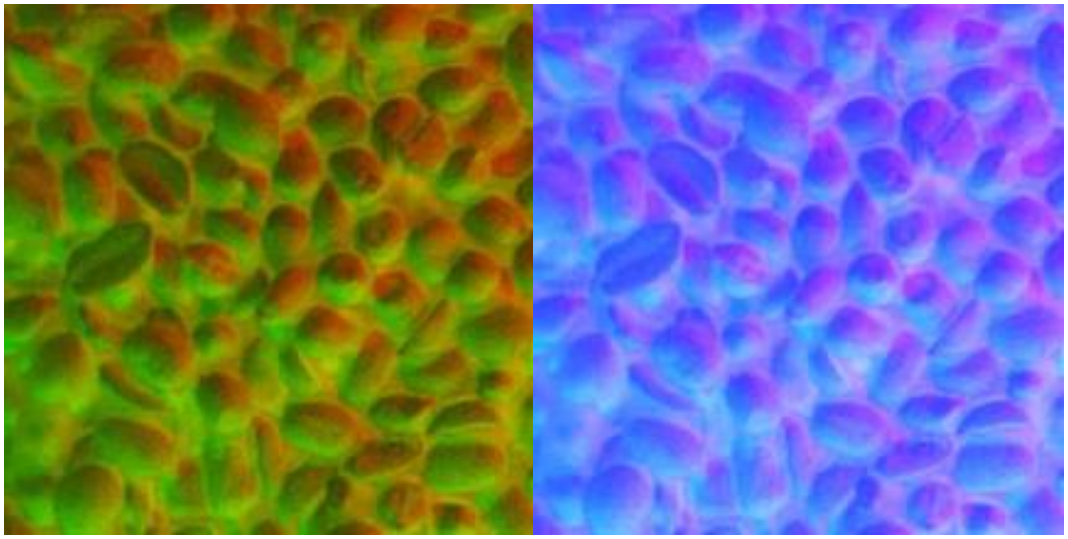
Ako je intenzitet slikovnog elementa teksture maske <=128

Intenzitet rezultata = (2 * intenzitet maske * intenzitet maskirajuće teksture) / 256.0

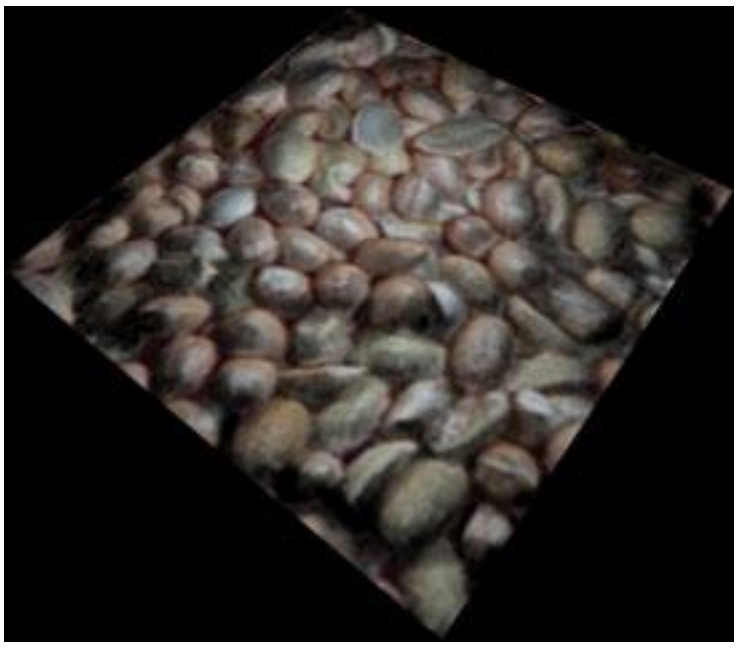
inače

intenzitet rezultata = 255 – ((255 – (2 * intenzitet maske – 128)) * (255 - intenzitet maskirajuće teksture)) / 256

Ovaj filtar funkcionira na način da propušta slikovne elemente maskirajuće teksture množeći njihove intenzitete s intenzitetima maske, no ne izravno, već s prilagođenjima, tako da rezultat bude nelinearno svjetliji gdje je maska svjetlija, isto tako za tamna područja. Lijevi prikaz slike 5.11 je rezultat primjene „overlay“ filtra na navedene teksture . Ukoliko se do sada nije računalo s plavom komponentom boje, dobiva se rezultat s lijevog prikaza. Budući da mapa normala tangencijalnog prostora ima plavu komponentu jednaku intenzitetu 128, a ova tehnika služi izradi tangencijalne mape normala, desni prikaz slike 5.11 za razliku od lijevog, sadrži intenzitet plave komponente jednak 128. Ukoliko se prati promjena plave komponente kroz postupak, rezultat bi također bio 128. Ovim korakom završava postupak izrade mape normala iz fotografija objekta pod različitim kutovima osvjetljenja. Slika 5.12 prikazuje primjenu dobivene mape normala preslikavanjem neravnina na model kvadrata, zajedno s teksturom difuzne komponente koja sadrži prirodne boje objekta.



Slika 5.11 – Prikaz rezultantne mape normala, s intenzitetom plave komponente jednakim 0 (lijevo) i 128 (desno)



Slika 5.12 – Prikaz primjene mape normala dobivene iz fotografija preslikavanjem neravnina

5.4.2. Izrada na temelju postojećih tekstura

Postoje dvije vrste mapa normala s obzirom na primjenu, a to su mape normala nekog materijala ili neravnina specifične površine, i mape normala specifičnih objekata. Prva vrsta mapa normala najčešće se izrađuje iz visinskih mapa nastalih proceduralnim algoritmima ili iz fotografija prirode, te primjenom različitih filtara u programima za manipuliranje slikama. Ta vrsta mapa normala je tema ovog načina izrade.

5.4.2.1. Izbor teksture

Pri izboru teksture za izradu mape normala vrijede ista pravila kao i pri izboru teksture za izradu mape normala. Ispravne fotografije su one koje predstavljaju površinu slikanu pod pravim kutem, uz ravnomjerno osvjetljenje, što manju perspektivnu distorziju, te bez većih objekata u sceni koji odudaraju od pozadine. Kao primjeri ispravnih i neispravnih slika mogu poslužiti teksture sa slike 5.6. Primjena filtra sivih razina na fotografiju je često prvi korak izrade mape normala na temelju postojećih tekstura. Sama primjena tog filtra je osrednje zadovoljavajuća reprezentacija visinske mape, no za mapu normala potrebno je izvući dodatne informacije.

5.4.2.2. Metoda gradijenta susjednih slikovnih elemenata

Kao što se može naslutiti iz imena ove metode, normala, odnosno boja za pojedini slikovni element rezultantne mape normala dobiva se razmatrajući intenzitete boja susjednih slikovnih elemenata.

Prvi način primjene ove metode fokusira se na neposredne susjede slijeva, odozgo, odozdo, i sa desne strane trenutnog slikovnog elementa za kojeg se računa normala. Korištene formule su zapanjujuće slične formulama za primjenu visinske mape tehnikom preslikavanja neravnina, u kojoj se također iz visinske mape trebao dobiti vektor normale na temelju gradijenta

susjeda. U nastavku je dan opis najjednostavnijeg postupka izrade mape normala od visinske mape.

Izradi se prazna tekstura veličine jednake teksturi visinske mape. Za svaki slikovni element visinske mape naprave se slijedeći koraci.

1. Za trenutni slikovni element scene dohvati okolne slikovne elemente visinske mape na temelju trenutnih UV koordinata. U ovom jednostavnoj primjeru, dohvaćaju se prvi susjedni slikovni elementi. S obzirom da su u bojama visinske mape iznosi intenziteta pojedinih kanala jednaki, uzima se vrijednost crvenog kanala.

IntenzitetGore = DohvatiBoju(visinskaMapa, UV.x,UV.y - 1).crvena;
IntenzitetDolje = DohvatiBoju(visinskaMapa, UV.x,UV.y + 1).crvena;
IntenzitetLijevo = DohvatiBoju(visinskaMapa, UV.x - 1,UV.y).crvena;
IntenzitetDesno = DohvatiBoju(visinskaMapa, UV.x + 1,UV.y).crvena;

2. Računa se gradijent intenziteta lijevog i desnog, te gornjeg i donjeg elementa, te se razlike pohrane kao vektori visinskih promjena u prostoru oko trenutnog ciljnog elementa. U ovom slučaju vektori su orijentirani kao rezultantna slika na ekranu, vrijednosti x osi rastu prema desno, vrijednosti y osi prema gore, a z osi prema promatraču, iz ekrana. Potrebno je naglasiti da se kao slobodni koeficijent, koji određuje značenje pojedinog gradijenta u smislu određivanja tangencijalne osi, postavlja željeni intenzitet mape normala. Željeni intenzitet može biti različit za x i y os, no ovdje se zbog jednostavnosti uzima isti iznos.

HorizontalnaRazlika = IntenzitetDesno – IntenzitetLijevo;
VertikalnaRazlika = IntenzitetGore – IntenzitetDolje;

VektorHorizontalnePromjene = (Intenzitet, 0, HorizontalnaRazlika);
VektorVertikalnePromjene = (Intenzitet, 0, VertikalnaRazlika);

3. Boja koja predstavlja vektor normale dobiva se normalizacijom vektorskog produkta dvaju vektora visinskih promjena.

NormalaIzVisinskeMape = VektorskiUmnožak(VektorHorizontalnePromjene, VektorVertikalnePromjene);

Dobiveni vektor može biti usmjeren u bilo kojem smjeru koji odgovara vektoru od ishodišta 3D prostora do neke točke jedinične sfere, ograničeno, naravno, kvantizacijom intenziteta visinske mape. Drugim riječima, ovime je dobiven vektor kojem vrijednosti sve tri komponente mogu biti u intervalu [-1,1]. Budući da se u teksturu ne može pohraniti boja čija komponenta ima intenzitet izvan intervala [0,1] (ili [0,255], ovisno o trenutno korištenom rasponu), potrebna je korekcija vektora. Kako bi se vektor ispravno preslikao iz raspona [-1,1] na raspon [0,1] , primjenjuje se jednostavna formula:

Rezultantni vektor = (vektor s komponentama raspona [-1,1]) / 2.0 + (0.5,0.5,0.5)

Rezultantni vektor očito poprima vrijednosti iz raspona [0,1] te se množenjem komponenata s faktorom 255 dobivaju boje koje se pohranjuju u odgovarajuća 24 bita svakog slikovnog elementa dobivene mape normala.

Kad bi se izvela vrijednost vektora koji se dobije prikazanim vektorskim umnoškom, prije vlastite normalizacije, došlo bi se do vektora:

(- Intenzitet * HorizontalnaRazlika, - Intenzitet * VertikalnaRazlika, Intenzitet * Intenzitet)

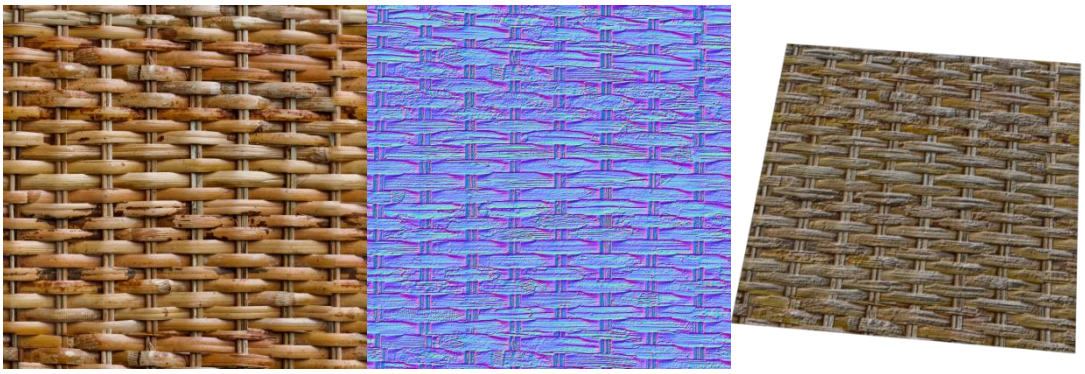
Što je po smjeru ekvivalentno vektoru:

(- HorizontalnaRazlika, - VertikalnaRazlika, Intenzitet)

Iz dobivenog vektora je vidljivo kako intenzitet utječe na otklon Z koordinate konačne normale. Nakon normalizacije, komponenta s prethodno najvećom vrijednosti ima još veći iznos u odnosu na ostale komponente. Zbog

prethodno navedenog, proizvoljni parametar u ovoj tehnici utječe izravno na intenzitet Z osi, te predstavlja otklon vektora normale od površine tangencijalnog prostora za taj slikovni element.

Slika 5.13 prikazuje izvornu fotografiju veziva pletene košare [21], mapu normala izrađenu metodom gradijenta te njenu primjenu na model kvadrata metodom preslikavanja neravnina.



Slika 5.13 – Prikaz izrade i primjene mape normala metodom gradijenta susjednih slikovnih elemenata

5.4.2.3. Metoda Sobelovog operatora

Sobelov operator je poznat po svojoj primjeni u metodama obrade slike, pogotovo metodama za određivanje ruba ili nagle promjene intenziteta slike. U svojoj osnovi, Sobelov operator koristi dvije matrice dimenzija 3x3 kojima se konvoluiraju slikovni elementi neke teksture kako bi se jače naglasile veće promjene intenziteta u susjednim slikovnim elementima slike.

Matrice Sobelovog operatora su

$$Gx = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \text{ i } Gy = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Jednostavna primjena matrica Sobelovog operatora na izradu mape normala iz visinske mape je očita: s obzirom na trenutno promatrani slikovni element, intenziteti njegovih susjeda pomnože se odgovarajućim težinskim faktorima matrica te se zbroje u nove vrijednosti koje predstavljaju težinske promjene

intenziteta po osima visinske mape. Naravno, budući da su to težinske sume, nakon zbrajanja podijele se sa sumom težina, kako bi raspon vrijednosti ostao u intervalu koji može predstavljati proizvoljni vektor normale. U nastavku su dane konkretne formule postupka.

IntenzitetGoreSredina = DohvatiBoju(visinskaMapa, UV.x,UV.y - 1).crvena;
IntenzitetDoljeSredina = DohvatiBoju(visinskaMapa, UV.x,UV.y + 1).crvena;
IntenzitetLijevoSredina = DohvatiBoju(visinskaMapa, UV.x - 1,UV.y).crvena;
IntenzitetDesnoSredina = DohvatiBoju(visinskaMapa, UV.x + 1,UV.y).crvena;
IntenzitetGoreLijevo = DohvatiBoju(visinskaMapa, UV.x - 1,UV.y - 1).crvena;
IntenzitetDoljeLijevo = DohvatiBoju(visinskaMapa, UV.x - 1,UV.y + 1).crvena;
IntenzitetGoreDesno = DohvatiBoju(visinskaMapa, UV.x + 1,UV.y - 1).crvena;
IntenzitetDoljeDesno = DohvatiBoju(visinskaMapa, UV.x + 1,UV.y + 1).crvena;

*HorizontalnaTežinskaPromjenaIntenziteta = (- IntenzitetGoreLijevo – 2 * IntenziteLijevoSredina – IntenzitetDoljeLijevo + IntenzitetGoreDesno + 2 * IntenzitetDesnoSredina + IntenzitetDoljeLijevo) / 8.0*

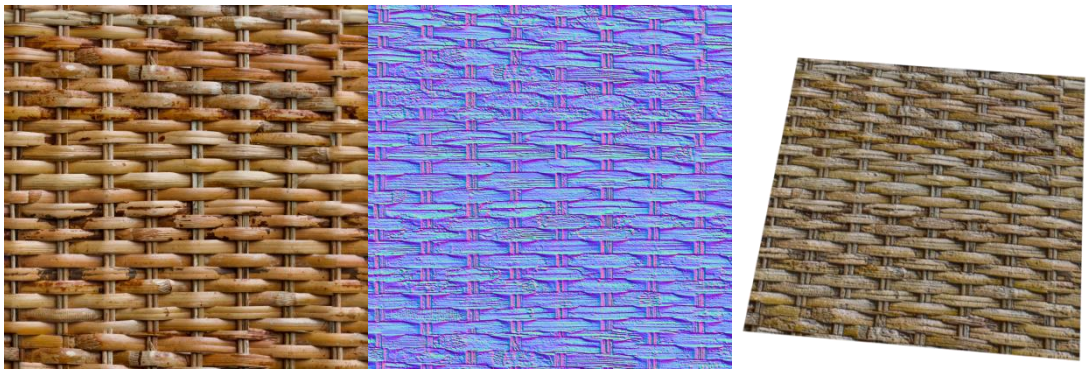
*VertikalnaTežinskaPromjenaIntenziteta = (- IntenzitetGoreLijevo – 2 * IntenziteGoreSredina – IntenzitetGoreDesno + IntenzitetDoljeLijevo + 2 * IntenzitetDoljeSredina + IntenzitetDoljeDesno) / 8.0*

Nakon izračuna horizontalne i vertikalne težinske promjene, potrebno ih je kombinirati u vektor na sličan način kao i u metodi gradijenta susjeda. Novi vektor koji predstavlja normalu ponovno ima parametarsku Z komponentu, koja je isključivo broj iz intervala [-128,127]. Vektor iznosi:
(*HorizontalnaTežinskaPromjenaIntenziteta,*
VertikalnaTežinskaPromjenaIntenziteta, ProizvoljniIntenzitet)

Kako bi se iz tog vektora dobila boja koja će se pohraniti na odgovarajući slikovni element mape normala, vektor se normalizira, što može rezultirati jediničnim vektorom bilo koje komponente iznosa iz raspona [-1,1], te se nad tim vektorom vrši korekcija kao i u prethodnoj metodi, formulom:

Rezultantni vektor = (vektor s komponentama raspona [-1,1]) / 2 + (0.5,0.5,0.5)

Kao što se može primjetiti, razlika ove dvije metode je neznatna, samo u težinama vrijednosti pojedinih susjeda, no ideja je ista: računanje promjene intenziteta slikovnih elemenata koji predstavlja elevaciju površine. Slika 5.14 prikazuje istu izvornu fotografiju veziva pletene košare, mapu normala izrađenu primjenom Sobelovog operatora te njenu primjenu na model kvadrata metodom preslikavanja neravnina.



Slika 5.14 – Prikaz izrade i primjene mape normala metodom primjene Sobelovog operatora slikovnih elemenata

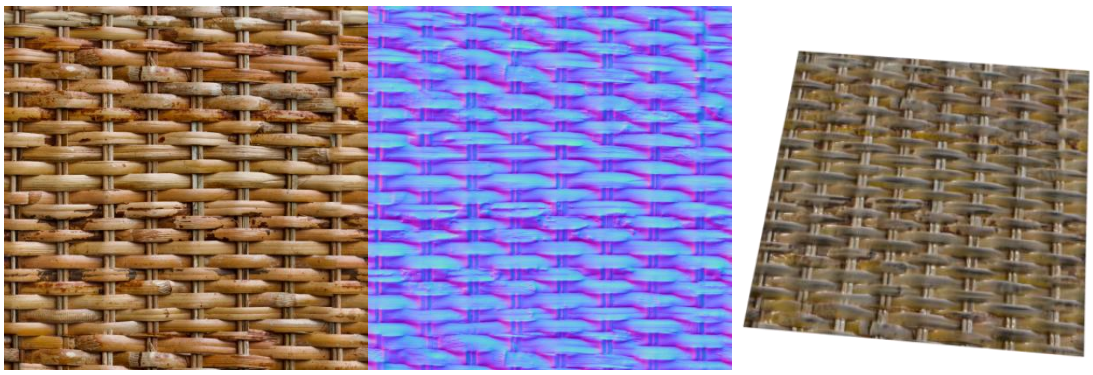
Kao što se može primjetiti iz mapa normala dobivenih upotrebom ove dvije tehnike, postoje vrlo šumoviti dijelovi teksture koji se preslikavaju na nagle promjene vektora i intenziteta osvjetljenja pri primjeni tih mapa normala, što nije uvijek željeni učinak. Jedna od mogućnosti je upotreba dodatnih filtara koji će zamućivati određene intenzitete slikovnih elemenata nekom jačinom, računanje vektora promjene intenziteta po osima pomoću matrica drugih

težina, ili uzimanje vrijednosti boja ne samo neposrednih susjeda, već i vrijednosti boja više ili manje udaljenih slikovnih elemenata.

5.4.3. Programska konverzija

Kao i u slučaju izrade visinskih mapa, razvila se potreba za programskom podrškom čija će se svrha isključivo sastojati od manipulacije i izrade mapa normala, kao i tekstura posebnih primjena u računalnoj grafici.

Ovdje se ponovno navodi program *Crazy Bump*, koji efikasno izrađuje i mape normala, i to iz visinske mape ili teksture u boji. Nitko osim tvoraca nije siguran kakvim algoritmima se dobivaju visoko kvalitetne mape normala iz slika koje nemaju pohranjenu informaciju o dubini, no primjena ovog i drugih programa za izradu mapa normala od postojećih tekstura je dovoljna za navođenje u ovom podpoglavlju. Zbog navedenih prednosti primjena mapa normala ispred visinskih mapa u metodi preslikavanja neravnina, programi poput *Crazy Bump*-a su iznimno cijenjeni i traženi, jer se mape normala tangencijalnog prostora danas intenzivno koriste, te je bitno imati programski alat kojim se ine izrađuju brzo i efikasno. Za kraj ovog podpoglavlja, u nastavku je dana slika 5.14, koja sadrži istu originalnu teksturu pletene košare, mapu normala dobivenu u programu *Crazy Bump*, te njenu primjenu na već poznati način.



Slika 5.14 – Prikaz izrade i primjene mape normala u programu *Crazy Bump*

5.4.4. Izrada mapa normala na osnovi 3D modela objekata

U podpoglavlju koje sadrži opis izrade visinskih mapa na osnovi 3D modela objekata spomenuto je kako većina programa za 3D modeliranje

sadrži različite mogućnosti preslikavanja svojstava jednog objekta na drugi objekt. Naravno da je među tim svojstvima prisutno i svojstvo preslikavanja geometrije jednog modela interpretirane kao vektori normala tih poligona na poligone drugog objekta.

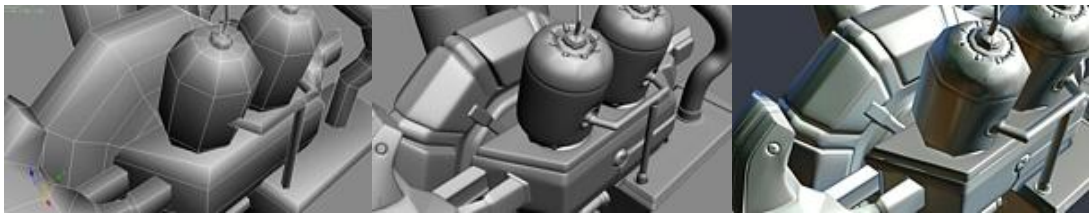
Kao i kod visinskih mapa, ovaj postupak je sličan za sve programe koji ga nude, te će u nastavku biti opisan općeniti postupak izrade mape normala, uz popratne slike koraka postupka napravljene u programu za modeliranje zvanom Modo.

Za početak, potrebno je izraditi model od kojeg se želi napraviti mapa normala. Slijedeće, potrebno je razmotriti svrhu mape normala koja će biti izrađena od modela. Postoje dvije osnovne praktične primjene mapa normala. Prva primjena je preslikavanje neravnina mapom normala na neke globalne površine u njihovom tangencijalnom prostoru, kao što su zidovi, stupovi, pod, zemlja, trava i druge statične površine. Mapa normala se u tim slučajevima izrađuje za proizvoljne površine koje ne moraju imati specifičnu mapu preslikavanja svojih poligona u UV prostor. Izrada ovakvih mapa normala od modela u praksi je vrlo rijetka jer je potrebna visoka razina detalja i velik broj poligona modela za izradu teksture koja se lagano mogla dobiti prethodnom metodom iz fotografije ili visinske mape.

Druga svrha upotrebe mapa normala je puno češća u praksi i iznimno ju je teško izvesti pravilno. Za izabrani objekt napravi se model niske rezolucije, koji će se koristiti kao izvor zraka, te model visoke rezolucije, čiju površinu će dosezati zrake koje će se računati iz modela niske rezolucije. Svrha ovog pristupa je ista kao u svim primjenama mape normala; dobiti vizualne detalje na što manjem broju poligona. Ovaj pristup nudi preslikavanje zakrivljenosti površine koja odgovara modelu na koji se primjenjuje, što je osnovna razlika od prethodno opisane primjene.

Druga razlika ovog pristupa je potreba za dva modela istog objekta, različite složenosti, no iste pozadinske kontstrukcije geometrije. Stručnjaci za izradu modela razvili su dvije osnovne tehnike za izradu dva potrebna modela. Prva tehnika je modelirati objekt niske rezolucije, pohraniti ga u

memoriju, i nastaviti ga mijenjati u model visoke rezolucije, no čuvajući općenitu strukturu. Druga tehnika pristupa problemu s druge strane: prvo se izrađuje model visoke rezolucije, pohrani u memoriju, a zatim se ili uklanjanjem detalja dobiva grublji model koji i dalje treba odgovarati obliku modela visoke rezolucije, ili se od početka gradi model niske rezolucije odgovarajućeg oblika. Slika 5.13 prikazuje dio modela motornog vozila niske rezolucije bez preslikavanja (lijevo), visoke rezolucije (sredina) i opet modela niske rezolucije s primjenom mape normala za preslikavanje površinskih neravnina [22]. Slijedi opis koraka za općeniti postupak izrade mape normala u 3D programu koristeći 3D modele objekata.

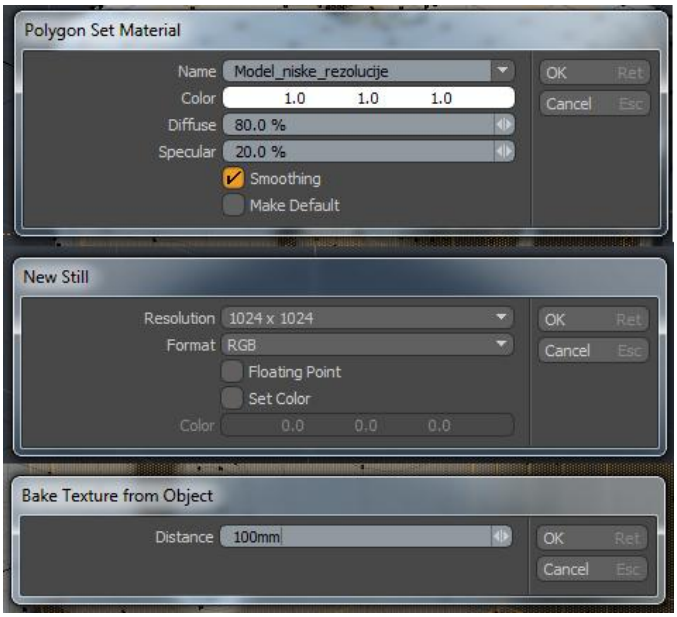


Slika 5.13 – Prikaz modela niske rezolucije bez i s mapom normala, te modela visoke rezolucije

Nakon učitavanja modela i pozicioniranja istih u prostoru scene, potrebno je izraditi UV mapu **samo za model niske rezolucije**, što je neizmjerno lakši posao od izrade kvalitetne UV mape za model visoke rezolucije. U nekim programima postoje funkcije koje ovisno o parametrima i obliku modela same generiraju UV mape za trokute modela.

Nakon stvaranja UV mape, potrebno je pridružiti materijal modelu niske rezolucije, te stvoriti praznu teksturu željene rezolucije u koju će biti pohranjen rezultat bacanja zrake, isto kao u procesu izrade visinske mape, s jednom razlikom: stvorena prazna tekstura za mapu normala mora imati komponente boje, dok je tekstura za visinsku mapu, koja je crno-bijela, mogla biti drugih formata, poput formata sivog spektra. Način korištenja za stvorenu teksturu se treba postaviti na preslikavanje neravnina mapom normala, što je ipak različito za pojedine 3D programe. Slijedi određivanje

parametara udaljenosti između modela, koji određuje duljinu zrake iz modela niske rezolucije. Slika 5.14 prikazuje navedene korake, s izuzetkom postavljanja načina korištenja teksture kao mapu normala.



Slika 5.14 - Koraci namještanja scene i postavki za izradu mape normala od modela postupkom „bacanja zrake“

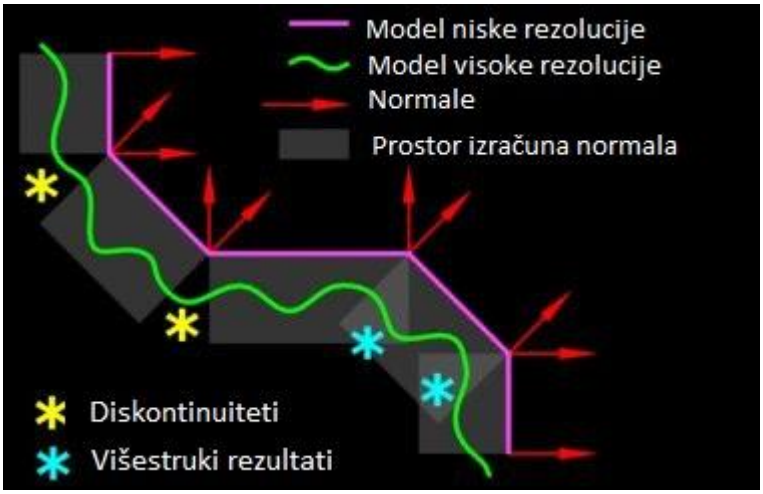
Parametar udaljenosti bacanja zrake treba iznositi udaljenost u trenutnim jedinicama prostora 3D programa koja odgovara najvećoj udaljenosti površine modela niske rezolucije do površine modela visoke rezolucije.

Sam postupak ispucavanja i praćenja zrake se vrši na način naveden u nastavku. Za svaki slikovni element rezultantne teksture određuje se prostorna koordinata trokuta modela niske rezolucije (koji je ovdje u ulozi projektivnog modela), koja odgovara položaju tog slikovnog elementa u prvom kvadrantu UV prostora u UV mapi tog modela. Dobivena prostorna koordinata je izvorišna točka „bacanja zrake“.

Zraka se ispuca iz izvorišne točke u smjeru normale točke na trokutu poligona koja se određuje interpolacijom vršnih točaka trokuta. Zraka

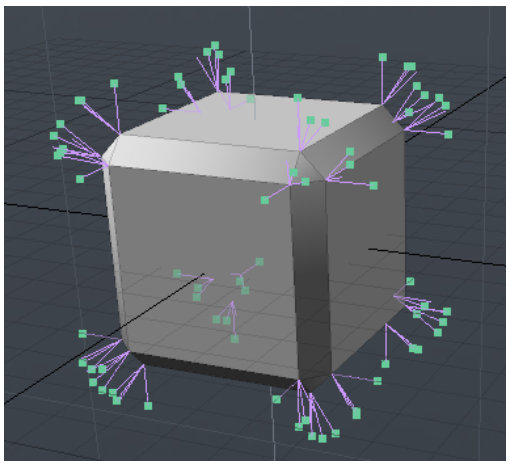
pravocrtno prelazi zadanu udaljenost, te se krene vraćati prema izvorišnoj točci u nadi da će naići na poligone modela visoke rezolucije. Ako i kada zraka u povratku presječe poligon modela visoke rezolucije, interpolirajući normale točaka koje su vrhovi presječenog poligona određuje se normala točke presjeka. Određena normala se prema prethodno navedenoj formuli za konverziju vektora normale pretvara u boju koja se zapisuje u odgovarajućem slikovnom elementu teksture za pohranu mape normala. Ukoliko za neki slikovni element zraka nije našla presjecište s geometrijom drugog modela, slikovnom elementu se pridružuje neutralna boja mape normala. Budući da se u ovom postupku izrađuje mapa normala tangencijalnog prostora, neutralna boja je ona koja prikazuje vektor (0,0,1) a to je boja (128,128,255).

Kao što se može primijetiti iz prošlog paragrafa, ovaj pristup je podložan velikim problemima koji proizlaze iz načina pohrane podataka o modelu. Jedan od najčešćih problema je taj da svaka točka modela može biti povezana s više normala, po jedna normala za svaki odgovarajući poligon čijih je ta točka sastavni dio. Kao što je prikazano na slici 5.15. [23], zbog višestrukih normala u točkama, iz pojedinih područja modela će se ispucati više zraka, čiji rezultati neće biti ispravni, već će dodatno djelovati na prethodne rezultate za isti slikovni element te pogoršati rezultantnu mapu normala. Na slici je također vidljiva suprotna situacija, gdje zbog rasporeda normala neke točke modela visoke rezolucije nikad neće biti pogođene, te ti detalji neće biti zapisani u bojama rezultantne mape normala. Za lakše razumijevanje slike prisutna je legenda za iscrtane dijelove.



Slika 5.15 – Prikaz problematičnosti pri višestrukim normalama u jednoj točci modela

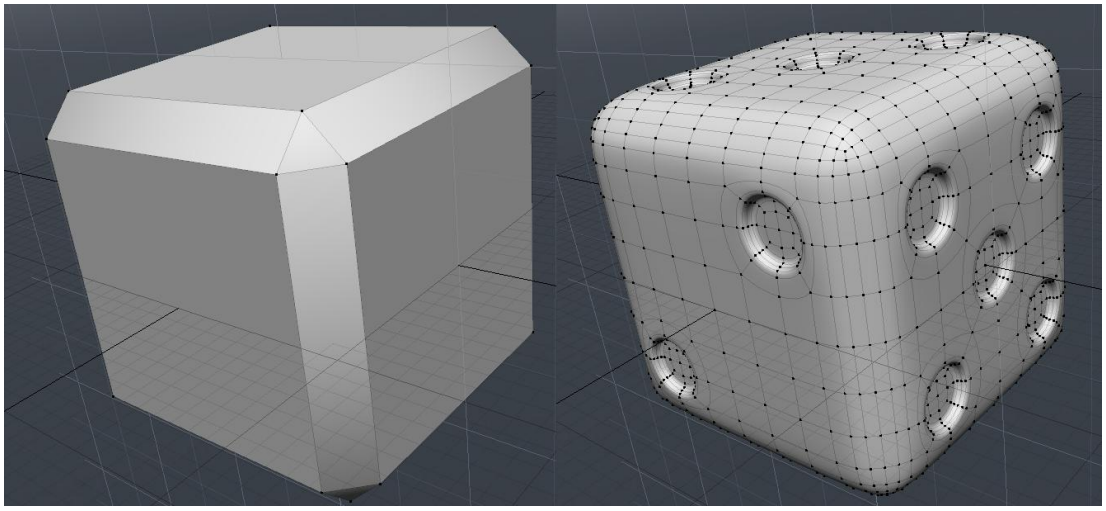
Kao još jedna razlika od izrade visinskih mapa tehnikom ispucavanja zrake iz jednog modela na drugi, u ovom pristupu je čest slučaj kao na slici 5.15, a to je da se zrake ispucavaju iz složenih često konkavnih tijela na druga konkavna tijela, dok se u izradi visinske mape najčešće koristi ravna ploha kao projekcijski model. Slika 5.16 prikazuje problem višestrukih normala kod jednostavnog modela kocke izbočenih stranica.



Slika 5.16 – Prikaz normala za svaku točku modela zbog ilustriranja problema višestrukih normala

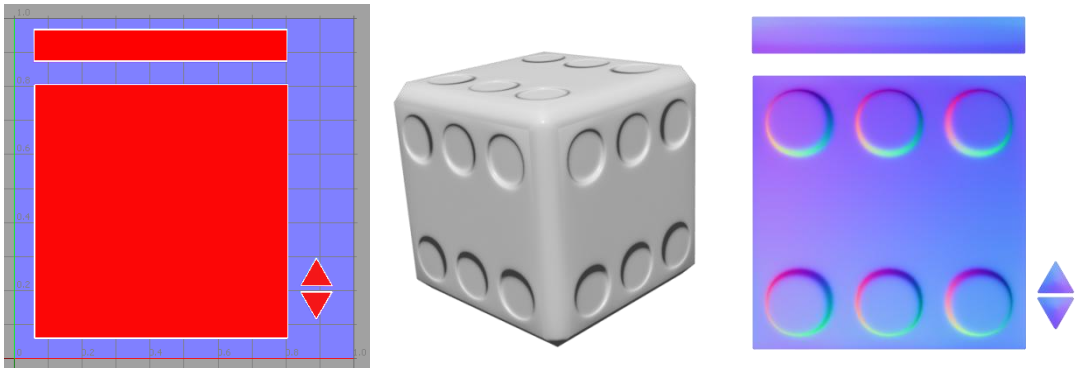
Sljedeći problem koji se javlja kod ove tehnike je organizacija UV mape modela. Ukoliko pojedini dijelovi modela niske rezolucije (poligoni)

dijele isti UV prostor, na slikovnim elementima mape normala koji odgovaraju tim koordinatama na UV mapi će ostati pohranjene normale od onog poligona niske rezolucije iz kojeg su kasnije ispucane i praćene zrake. Ukoliko je model simetričan, postoje posebne tehnike koje omogućavaju maksimalno iskorištenje UV prostora uz minimalne ili nepostojeće artefakte na mapi normala. Slika 5.17 prikazuje model niske rezolucije (lijevo) i model visoke rezolucije (desno) koji će se koristiti u nastavku opisa ove tehnike.



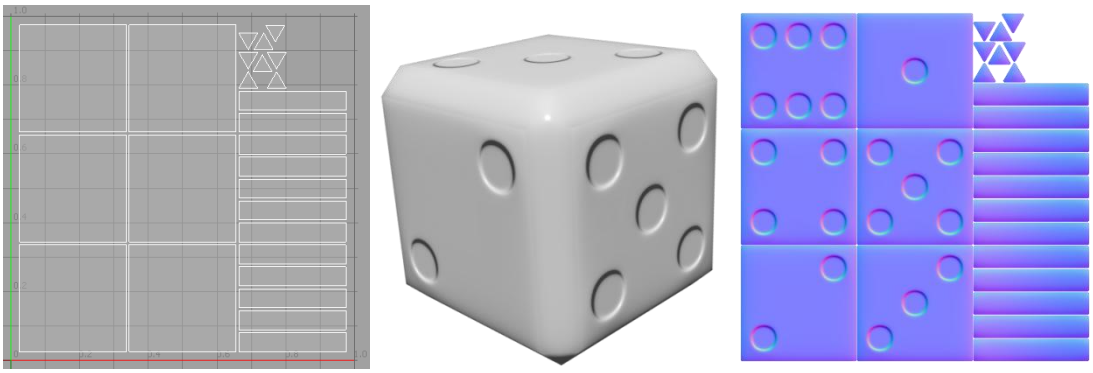
Slika 5.17 – Prikaz modela niske rezolucije i modela visoke rezolucije

Slika 5.18 prikazuje izradu mape normala pomoću dva gornja modela, no uz izraženu potrebu da se maksimalno iskoristi UV prostor, što je rezultiralo, naravno, krivom mapom normala. Očito je da se radi o preglatkom modelu obične igraće kocke. Potrebno je primjetiti kako su na slici 5.17 vidljive stranice kocke s brojevima 2, 3 i 5 (točnije s tim brojem točaka), a srednji prikaz na slici 5.18, kao i mapa normala, prikazuju samo stranicu s brojem 6. Očito je da je ta stranica modela, odnosno da su trokuti koji čine tu stranicu bili zadnji u redoslijedu za ispucavanje zrake, te su sve prethodne informacije slikovnih elemenata prebrisane s tim vrijednostima.



Slika 5.18 – Prikaz neispravne UV mape za izradu mape normala modela, izrađene mape normala, te primjena mape normala na model niske rezolucije

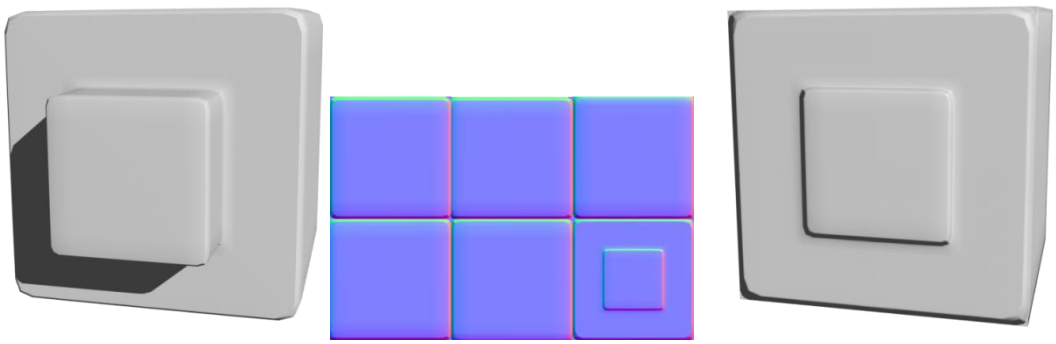
Ispravno UV mapiranje se dobiva odvajanjem svih poligona modela na način da svaki od njih dobije vlastiti UV prostor. Slika 5.19 prikazuje ispravnu UV mapu modela, dobivenu mapu normala, te izgled modela niske rezolucije s primjenom te mape za preslikavanje neravnina.



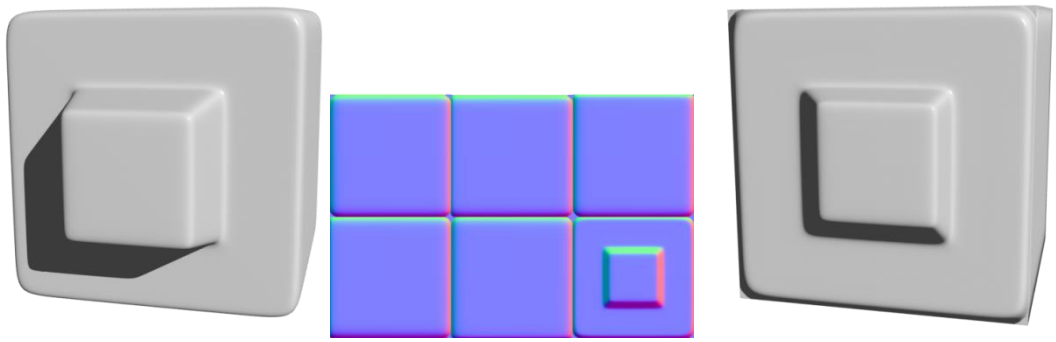
Slika 5.19 – Prikaz ispravne UV mape za izradu mape normala modela, izrađene mape normala, te primjena mape normala na model niske rezolucije

Potrebno je spomenuti još jedan problem u izradi mapa normala modela visoke i niske rezolucije. Budući da se ispucane zrake pomiču pravocrtno, kako bi se dobio gladak prijelaz smjera vektora normala na izrađenoj mapi normala, oba modela moraju imati međusobno nakošene poligone, te ne smije biti oštrih i naglih prijelaza između nagiba susjednih poligona. Ukoliko postoje nagle promjene nagiba, rezultantna mapa normala neće sadržavati dovoljno informacija za ispravni prikaz rezultata. Slika 5.20

prikazuje model visoke rezolucije s naglim nagibom (lijevo), dobivenu mapu normala ispucavanjem zrake, te model niske rezolucije (obična kocka) pomoću kojeg je rađena i na kojem je primjenjena mapa normala.



Slika 5.20 – Prikaz modela s naglim nagibom između poligona kao loš primjer modela visoke rezolucije za izradu mape normala



Slika 5.21 - Prikaz modela s blažim nagibom između poligona kao bolji primjer modela visoke rezolucije za izradu mape normala

Na slici 5.21 vidljiv je blaži nagib između poligona u prednjem izbočenju modela kocke visoke rezolucije, razlika u strmini prijelaza mape normala te bolji konačni rezultat primjene mape normala nego na prikazima sa slike 5.20.

Nakon navođenja svih problematičnosti potrebno je napomenuti da ova metoda, iako naporna, daje veličanstvene rezultate koji se redovito primjenjuju u industriji računalnih igara, kako bi modeli likova i živih bića,

kipova, vozila i predmeta svih veličina izgledali što realnije, sa što manjim brojem poligona za bolje performanse. Ispravno izrađene i primjenjene mape normala pružaju tu beneficiju.

5.4.5. Preuzimanje postojećih mapa normala s Interneta

Za razliku od visinskih mapa, na Internetu postoji daleko manji broj stranica s dostupnim teksturama mapa normala, jer način njihovog dobivanja i značenja nije poznat i intuitivan kao značenje visinske mape. Dodatni razlog, koji je neznatan u odnosu na nedostupnost mapa normala zbog nedovoljnog znanja je taj što se visinske mape mogu lagano dobiti čak i filtrom sivog spektra ili bojanjem prazne teksture bojama iz sivog spektra, dok je ručna izrada kvalitetne mape normala bojanjem svih slikovnih elemenata iznimno težak posao. U traženju mapa normala na Internetu bolje rezultate će pružiti potraga za što jeftinijim programskim alatom za izradu mape normala iz fotografija ili visinskih mapa.

5.5. Načini primjene mape normala

U praksi postoji više načina primjene mape normala, no velika većina odlazi upravo na metodu preslikavanja neravnina. Postoje i druge primjene, u kojima se boje mapa normala također interpretiraju i koriste kao vektori, no za dinamiku i kinematiku tijela, a ne kao dio izračuna intenziteta osvjetljenja. Ovo poglavlje ima veći naglasak na prvu primjenu, dok su druge površno spomenute i ostavljene za daljnja istraživanja.

5.5.1. Preslikavanje neravnina korištenjem mapa normala

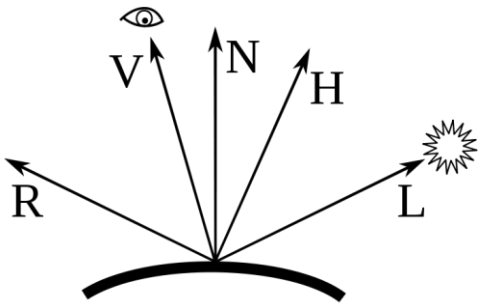
Preslikavanje neravnina korištenjem mapa normala je postupak u kojem se podaci iz teksture primjenjuju u izračunu intenziteta osvjetljenja modela umjerenog broja poligona, kako bi se postigao efekt površinskih neravnina za koje bi bila potrebna prevelika količina poligona s obzirom na malenu vidljivost detalja. U primjeni mapa normala, originalni vektor normale na neki slikovni element se u grafičkom protočnom sustavu potpuno zamjenjuje normalom određenom iz boje preuzete iz odgovarajućeg slikovnog elementa mape normala za određene koordinate UV mape modela. Primjena ove metode zahtjeva mali broj računskih operacija i jednu operaciju dohvaćanja vrijednosti iz spremnika teksture, te se jednostavno implementira u bilo kojem modelu sjenčanja. Vizualni rezultat ove metode je kvalitetniji od primjene preslikavanja neravnina korištenjem visinske mape, jer u ovom pristupu rezultat izračuna osvjetljenja ne ovisi o normalama geometrije na koju se primjenjuje tekstura, već isključivo o podacima preuzetim iz mape normala, što daje realističniji dojam da sjene na objektu bolje „prate“ pomak svjetla. Kako ni u ovom pristupu ne postoji stvaranje dodatne geometrije u protočnom sustavu, a dobivaju se bolji rezultati od korištenja visinske mape za istu tehniku, ovaj način upotrebe mapa normala za izračun intenziteta osvjetljenja je postao prevladavajući u svijetu.

Slijedi kratki pseudokod korištenja mape normala za preslikavanje neravnina korištenjem Phongovog modela sjenčanja, uz jedno usmjereno

svjetlo prisutno u sceni. Ulazni podaci su: UV koordinata trenutnog slikovnog elementa (u nastavku UV), položaj trenutne točke modela u sceni (u nastavku PoložajTočke), položaj svjetla u sceni (u nastavku PoložajSvjetla), vektor od očišta do trenutne točke modela (u nastavku VektorPogleda).

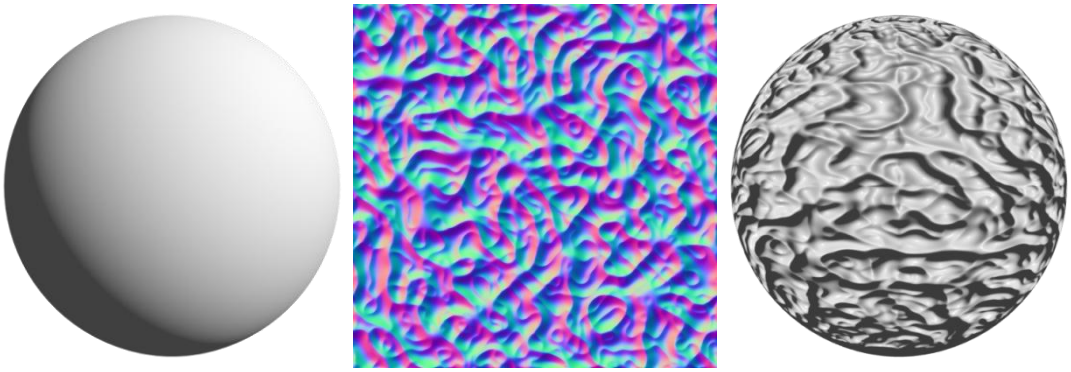
```
BojaNormale = DohvatiSlikovniElementIzTeksture (mapa normala, UV);
Normala = 2 * (BojaNormale – (128,128,128)) / 256.0;
VektorSvjetla = Normaliziraj (PoložajSvjetla – PoložajTočke)
DifuznaKomponenta = Minimum (skalarniProdukt (Normala, VektorSvjetla), 0);
ReflektiraniVektor = 2 * SkalarniProdukt (Normala, VektorSvjetla) * Normala – VektorSvjetla;
ReflektivnaKomponenta = Potenciraj ( SkalarniUmnožak (ReflektiraniVektor, VektorSvjetla),
potencijaReflektivneKomponente);
```

Slika 5.22 ilustrira vektore korištene u Blinn-Phongovom modelu sjenčanja. Vektor označen slovom N je normala trenutne točke modela, L je vektor od te točke do izvora svjetlosti, R je vektor L reflektiran oko normale, V je vektor od točke modela do očišta, a vektor H zvan poluVektor se koristi za aproksimaciju izračuna kuta između vektora R i N



Slika 5.22 – Prikaz potrebnih vektora za razumjevanje Blinn-Phongovog modela sjenčanja [32]

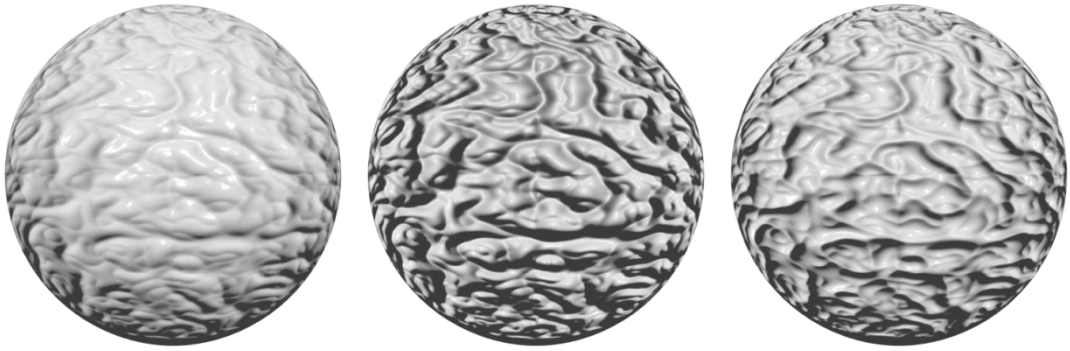
Kao što se može primjetiti, u ovom modelu sjenčanja normala dobivena iz mape normala koristi se za izračunavanje difuzne i reflektivne komponente rezultantne boje slikovnog elementa. Slika 5.23 prikazuje model kugle bez primjene mape normala, mapu normala te njenu primjenu na model korištenjem Phongovog modela sjenčanja.



Slika 5.23 Prikaz primjene mape normala na preslikavanje neravnina

Kao što je vidljivo iz prethodnog pseudokoda, primjena mapa normala je puno jednostavnija od primjene visinske mape za istu tehniku, jer ovdje nije potrebno računanje normale iz vrijednosti susjednih slikovnih elemenata, manje je instrukcija, jednostavnije je i brže, no manja je kontrola u primjeni.

Iz poglavlja „Primjena visinskih mapa“ poznato je kako za vrijeme izvođenja možemo mijenjati parametar koji određuje utjecaj visinske mape na normalu geometrije na koju se primjenjuje, kao i parametar jakosti Z komponente normale. Kod mapa normala ti parametri ne postoje, te se kontrola nad njihovim intenzitetom postiže mijenjanjem vektora koji treba nakon promjena i dalje biti jediničan, a rezultati tih promjena nisu predvidljivi kao kod visinskih mapa. Ono što se jednostavno može postići jest inverzija intenziteta pojedinih kanala boje (što nije preporučljivo za kanal plave boje), no to može uzrokovati neispravan izgled sjena s obzirom na odnos položaja modela i izvora svjetla. Ukratko, intenzitet mapa normala može se mijenjati pri izradi, pa se odabir radi između više tekstura različitih intenziteta. Slika 5.24 prikazuje primjene 3 mape normala na model kugle. Mape normala su izrađene od iste visinske mape u programu Crazy Bump primjenom različitih iznosa parametra intenziteta.



Slika 5.24 – Prikaz primjene mape normala različitih intenziteta

Spomenuto je kako primjena mapa normala ima realističniji utjecaj na promjenu kuta osvjetljenja. Slika 5.25 ilustrira dvije primjene mape normala s različitim kutem osvjetljenja, točnije: iznosom rotacije usmjerenog svjetla u sceni.

Slika 5.25 – Prikaz utjecaja promjene kuta osvjetljenja na primjenu mape normala

Mapa normala se za preslikavanje neravnina koristila pretežno samostalno, no kroz razvoj tehnika preslikavanja zaklanjanja uslijed paralakse ili promjene pogleda (engl. *parallax mapping*) većinska upotreba mapa normala za preslikavanje neravnina sve više naginje toj tehnici. Obje tehnike primjene mapa normala koriste se i u čestičnim sustavima gdje je svaka čestica predstavljena kvadratom uvijek orijentiranim prema očištu, te se sjenčanjem uz pomoć mape normala postiže realističnost prikaza, tako da promatrač nema dojam da se radi o kvadratima umjesto 3D modelima.

5.5.2. Primjena u čestičnim sustavima

Pojam mapa normala se općenito shvaća kao tekstura s bojama koje se interpretiraju kao vektori normala geometrije na koje će se primjeniti pojedina mapa normala. No postoji i drugi pristup mapa normala koji pretpostavlja primjenu teksture kao izvora podataka bez geometrije na koju se tekstura projicira UV mapom modela. U tom slučaju mape normala pohranjuju vektore smjera kretanja pojedinih čestica ili vektore refleksije od neke površine. Takva primjena mapa normala naziva se simulacijom sustava čestica i detekcije kolizije baziranom na grafičkim računalnim komponentama, iako je to „šire shvaćeni“ pojam mapa normala, koji se rijetko koristi u praksi.

6. Aplikacija Normal Maker

Kao drugi praktični dio ovog završnog rada izrađena je programska aplikacija nazvana Normal Maker, nazvana tako zbog funkcionalnosti učitavanja i procesiranja postojećih tekstura u svrhu dobivanja mape normala koja se može pohraniti u memoriju i dalje koristiti u bilo kojoj od primjena navedenih u prethodnom poglavlju.

6.1. Korištena tehnologija

Za izradu ove aplikacije ponovno je korištena Microsoft XNA tehnologija zbog istog razloga, jednostavnog korištenja postojeće bibilioteke, koja dolazi zajedno s navedenom tehnologijom, u svrhu implementacije jednostavnih efekata i filtara koji mijenjaju slikovne elemente teksture na određeni način. Programi za sjenčanje slikovnih elemenata pisani su u jeziku HLSL (engl. High level shader language), čije naredbe se prevode i šalju grafičkoj kartici pomoću programskog sučelja Microsoft Direct X. Kao i u aplikaciji Height Maker, ostvarena je integracija XNA tehnologije sa Windows

Forms projektom, ponovno zbog jednostavnosti organizacije i implementacije rada korisničkog sučelja. Aplikacija je pisana u programskom jeziku C#.

6.2. Logika i protočni sustav aplikacije

Za glavni tok logike aplikacije se koristi klasa Game.cs iz biblioteke Microsoft.Xna.Framework. Kao što je spomenuto u logici i protočnom sustavu aplikacije Height Maker, klasu game čine dvije metode koje se pozivaju pri inicijalizaciji (Initialize() i Load()) te jedna metoda koja se poziva pri zatvaranju aplikacije (Unload()). Ostatak vremena naizmjenično se pozivaju 2 metode: metoda Update() za promjenu podataka logike i metoda Draw() za crtanje grafičkog sadržaja na ekranu kojeg stvara Game klasa.

Korištenjem klase Form iz biblioteke System.Windows.Forms postiže se prikazivanje drugog prozora čija uloga je prikaz korisničkog sučelja.

Korisničko sučelje ove aplikacije sadrži mogućnosti učitavanja teksture, spremanja trenutno obrađivane teksture u memoriju, primjenu efekata koji mijenjaju intenzitete boje slikovnih elemenata, promjenu njihovih parametara te obnavljanje trenutno obrađivane teksture na originalnu teksturu, čiji se sadržaj u memoriji ne može promijeniti korištenjem ove aplikacije.

Trenutačno podržani ulazni formati tekstura su .jpg, .png i .tif, isključivo iz razloga lako dostupnih funkcija biblioteka koje učitavaju te vrste datoteka u prikladni memorijski objekt kojeg grafički dio implementacije koristi za slanje tekstura grafičkoj kartici. Zbog jednostavnosti i kvalitete tog formata pohrane tekstura, te većinsku podržanost tog formata u svim aplikacijama za manipulaciju slikama, jedini podržani izlazni format je, kao i u slučaju aplikacije Height Maker, format png.

6.2.1. Detalji implementacije manipulacije teksturama

Kao što je svaki prikaz na ekranu samo projekcija 3D točaka, tako je i prikaz teksture preslikavanje 3D prostora scene na 2D prostor ekrana. Kao

što je objašnjeno u poglavlju „Opis modela kao skupa podataka“, svaki model je skup točaka s različitim parametrima. Minimalni broj točaka za prikazivanje pravokutne teksture na ekranu je 4. Te točke čine dva trokuta. Za prikazivanje teksture točkama potrebno je postaviti 3D koordinate, kako bi se znao njihov položaj u prostoru scene, UV koordinate točaka, da grafička kartica zna kako se tekstura preslikava na trokute koje čine te točke, te normale točaka, da se odredi smjer u kojem su okrenuti trokuti, te da ih se crta okrenute na tu stranu. Za primjenu u ovoj aplikaciji, položaji točaka su postavljeni na točke (-0.5,0,-0.5) (0.5,0,-0.5) (-0.5,0,0.5), (0.5,0,0.5), što odgovara kvadratu duljine stranice 1, koji leži na XZ osi. Normala za sve 4 točke je postavljena na (0,1,0). Koordinate UV mape postavljene su na način da ispunjavaju cijeli 1. kvadrant UV prostora: (0,0), (0,1), (1,0), (1,1).

XNA grafička biblioteka sadrži funkciju koja je korištena za crtanje teksture na ekran. Funkcija s općenitim nazivom parametara prikazana je u nastavku.

graphicsDevice.DrawUserIndexedPrimitives< deklaracija podataka o točci >(Tip primitiva za crtanje, niz podataka za svaku točku, indeks početne točke za crtanje, broj točaka za crtanje, niz prirodnih brojeva koji predstavljaju indeksirane točke u određenom redoslijedu, početni indeks, broj primitiva za crtanje);

Kao što je vidljivo iz prvog reda naredbe, za ovaj način slanja točaka grafičkoj kartici na crtanje, potrebna je deklaracija podataka o točci. Deklaracija sadrži zapis podataka za jednu točku i određenom redoslijedu, te zapis veličine i pomaka pojedinog podatka u odnosu na početak podataka o točci. U ovoj aplikaciji korištena je postojeća deklaracija iz XNA grafičkih biblioteka: VertexPositionNormalTexture, koja sadrži sve potrebne podatke za ovu primjenu.

Budući da dane točke čine model kvadrata, one se moraju transformirati matricom modela, pogleda i projekcije kako bi bile preslikane na ravninu ekrana. Budući da je u ovoj aplikaciji ideja da ekran uvijek bude ispunjen teksturom koja se obrađuje, ne koristi se matrica pogleda, što je ekvivalentno jediničnoj matrici u koja je neutralni element za matrično množenje. Kako model uvijek ostaje iste veličine, položaja i rotacije kako bi pružao potpuni pogled na teksturu, ne koristi se niti matrica modela.

Matrica projekcije se, za razliku od prethodnih, mora zadati jer ne postoji implicitan način preslikavanja 3D točaka u 2D ravninu ekrana. Kako se htio postići ispravan način prikazivanja, za matricu projekcije koristi se ortografska matrica, kao i u aplikaciji Height Maker.

No za razliku od te aplikacije u kojoj je prozor za pregled modela stvoren XNA Game klasom, ova aplikacija mijenja veličinu prozora ovisno o rezoluciji učitane teksture za obradu, kako bi pri njenoj obradi prikaz bio u stvarnom omjeru veličina teksture.

Slijedi kod koji se poziva pri svakom učitavanju teksture za obradu, te služi za prilagodbu veličine prozora.

```
public void SetBackBufferToFitScaledTextureFully(Texture2D textureToSize)
      {
         Point displaySize = new
Point(_graphicsDeviceManager.GraphicsDevice.DisplayMode.Width,
           _graphicsDeviceManager.GraphicsDevice.DisplayMode.Height);

         Point textureSize = new Point(textureToSize.Width, textureToSize.Height);

         if (textureSize.X <= displaySize.X && textureSize.Y <= displaySize.Y)
         {
            SetBackBufferToPointSize(textureSize);
            return;
         }

         displaySize.X = (displaySize.Y * textureSize.X) / textureSize.Y;
         displaySize.Y = (displaySize.X * textureSize.Y) / textureSize.X;
         textureSize = displaySize;

         SetBackBufferToPointSize(textureSize);
      }
```

Prikazani kod samo provjerava je li veličina teksture manja u obje osi od ekrana, te ako je, prilagođava veličinu ekrana na veličinu teksture. Ako je slučajno jedna od osi veća nego trenutna veličina ekrana, pronalazi se najveća veličina ekrana koja je u proporcijama učitane teksture, a to se postiže jednostavnim skaliranjem veličina obje osi ekrana s omjerom veličina osi teksture.

Bez obzira na veličinu ekrana, ortografska matrica se u svakom slučaju definira tako da se lijeva, desna, gornja i donja ravnina ekrana

postave tako da odgovaraju točno udaljenostima točaka ekrana od sredine ekrana, kako bi konačni prikaz teksture bio projiciran točno na ekran.

Kako je ova aplikacija više usmjerena na rad s teksturama, bilo bi prikladno opisati ukratko korištene klase i njihova svojstva. Korištene su dvije različite klase, klasa Texture2D i klasa RenderTarget2D.

Klasa Texture2D služi za pohranu teksture, te njeno slanje u grafički protočni sustav tako da se postavi kao izvor slikovnih elemenata u parametru programa za sjenčanje. Klasa sadrži osnovna potrebna svojstva za manipulaciju slikovnim elementima, a to su dohvat dimenzija teksture, dohvat svih slikovnih elemenata teksture i pohranu svih elemenata. Zbog neprilagođenosti zadnja dva svojstva, primjena operacija nad svim slikovnim elementima teksture nije prikladna za središnji procesor računala, jer za promjenu proizvoljnog broja slikovnih elemenata treba dohvaćati sve elemente, iterirati po njima, te ih pohraniti nazad u teksturu, što je vrlo teško izvedivo 60 puta u sekundi s većim teksturama. Prva implementacija ove aplikacije je pokušala koristiti središnji procesor i klase Texture2D za primjenu efekata na slikovne elemente, te je bila prespora za jednu primjenu, što je daleko od 60 primjena u sekundi.

U grafičkom protočnom sustavu postoje razni spremnici podataka poput spremnika podataka za čitanje tekstura, ulazno-izlazni spremnik ispunjen podacima o točkama koje se crtaju, dubinski spremnik, spremnik konačnog prikaza na ekran i drugi.

Funkcije iz XNA grafičke biblioteke u suradnji s Direct X programskim sučeljem omogućavaju postavljanje objekta tipa RenderTarget2D kao ciljnog izlaznog spremnika za pohranu slikovnih elemenata koji su izračunati za prikaz na ekranu. Budući da klasa RenderTarget2D nasljeđuje implementaciju klase Texture2D, moguće je postaviti referencu na objekt tipa Texture2D da pokazuje na objekt tipa RenderTarget2D u kojem je upravo

pohranjen izračun scene za prikaz teksture na ekranu. Na tom principu se temelji rad ove aplikacije.

Nakon objašnjenja korištenih podataka i načina prikaza teksture na ekranu, slijedi opis rada aplikacije na apstraktnoj razini.

Aplikacija sadrži 3 glavne varijable u kojima se pohranjuju teksture. Prva varijabla zove se „originalTexture“, te sadrži referencu na kopiju teksture koja je učitana u aplikaciju, a nalazi se u radnoj memoriji. Ta varijabla koristi se za obnavljanje radne teksture ukoliko ju se promijeni efektima na neželjeni način. Radna tekstura je varijabla „workingTexture“. U nju pohrani tekstura prilikom učitavanja, te služi kao odredište kopiranja originalne teksture u slučaju obnavljanja teksture. Tekstura se naziva radnom zato što svaka primjena efekta mijenja isključivo radnu teksturu. Treća varijabla je „displayTexture“, koja se koristi kao tekstura za prikaz te se crta na ekranu prije i nakon svake primjene efekta. Potrebno je napomenuti dvije ključne stvari vezane uz ovu varijablu: tekstura za prikaz je rezultat primjene trenutno aktivnog efekta na radnu teksturu, te je ova tekstura izvor podataka za pohranu u memoriju pri odabiru te opcije.

Ukratko, učitavanje nove teksture postavlja sve tri varijable teksture, primjena efekta mijenja radnu teksturu, koja potom mijenja teksturu za prikaz, a obnavljanje teksture postavlja radnu teksturu na izgled originalne teksture.

Kako bi se postigla mogućnost izrade mapa normala u ovoj aplikaciji, implementirani su efekti koji se mogu neograničeno primjenjivati na svaku učitanu teksturu.

Prvi implementirani efekt ekvivalentan je primjeni filtra sivog spektra, čija primjena je objašnjena u prethodnim poglavljima. Implementirani efekti koji pretvaraju postojeću teksturu u mapu normala nakon primjene filtra sivog spektra su efekt koji primjenjuje metodu gradijenata susjednih slikovnih

elemenata te efekt koji konvoluira intenzitete slikovnih elemenata primjenom matrica Sobelovog operatora. Oba efekta imaju dodatnu opciju odabira intenziteta Z komponente boje, a svi detalji su već opisani u podpoglavlju „Načini izrade mapa normala“.

Kako je pokazano u istom poglavlju da program Crazy Bump daje daleko bolje rezultate koji imaju manje alterniranja u intenzitetima slikovnih elemenata nego rezultati dobiveni prethodno navedenim metodama, u ovoj aplikaciji implementirana su dva jednostavna efekta čija namjena je smanjenje razlike intenziteta slikovnih elemenata.

Prvi takav efekt je efekt srednjeg elementa (engl. Median filter). Algoritam koji se primjenjuje na slikovne elemente je kratak:

Za trenutni slikovni element dohvate se susjedni slikovni elementi. Odredi se srednji intenzitet od svih intenziteta prisutnih u dohvaćenim slikovnim elementima za svaku komponentu boje. Srednji intenzitet je potrebno razlikovati od aritmetičke sredine elemenata; srednji intenzitet jest jedan od elemenata, koji može biti daleko od aritmetičke sredine niza, a rezultat aritmetičke sredine ne mora odgovarati vrijednosti nekog elementa. Komponente koje odgovaraju srednjim vrijednostima čine rezultantnu boju tog slikovnog elementa. Slika 6.1 [24] prikazuje rezultat (desno) višestruke primjene efekta srednjeg elementa na originalnu teksturu (lijevo)

Slika 6.1 – Prikaz originala i rezultata primjene efekta srednjeg elementa

Drugi efekt implementiran za smanjenje razlike u intenzitetima slikovnih elementa je Gaussov filtar ili Gaussovo zamućenje (engl. Gaussian blur). Temelji se na sumiranju umnožaka intenziteta susjednih slikovnih elemenata pomnoženih s težinskim faktorima koji odgovaraju Gaussovoj krivulji. Parametar koji je dio implementacije ovog filtra je radijus ili udaljenost susjednih elemenata koji se uzimaju u obzir za izračun konačnog intenziteta trenutnog slikovnog elementa te se množe s težinskim faktorima. Slika 6.2 prikazuje primjenu Gaussovog filtra na teksturu, i to primjene s 2 različita radijusa. Lijevi prikaz je originalna tekstura, srednji prikaz prikazuje original nakon primjene zamućenja radijusa 5 slikovnih elemenata

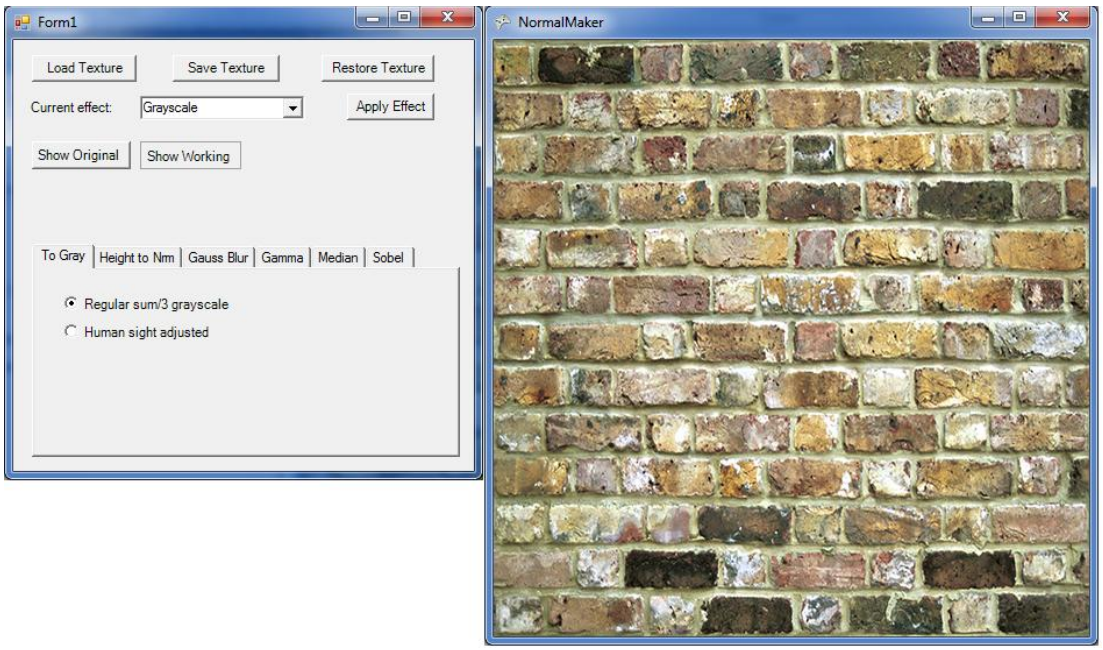
Slika 6.2 – Prikaz rezultata primjene Gaussovog filtra na teksturu u aplikaciji Normal Maker

6.3. Upute za korištenje aplikacije

Aplikacija omogućava učitavanje teksture iz memorije tijekom izvođenja te osnovnu manipulaciju nad slikovnim elementima pomoću par efekata. U bilo kojem trenutku izvođenja moguće je spremiti trenutačni prikaz teksture u memoriju.

Pri pokretanju aplikacije otvaraju se dva prozora, jedan sadrži korisničko sučelje aplikacije, drugi se koristi za iscrtavanje teksture u 3D prostoru projekcijom na 2D prostor.

Odmah nakon pokretanja aplikacije moguće je učitati postojeću teksturu te mijenjati njenu kopiju u radnoj memoriji. Slika 6.3 prikazuje izgled oba ekrana nakon pokretanja programa i učitavanja proizvoljne postojeće teksture (jer niti jedna tekstura nije učitana pri pokretanju i ekran za prikaz teksture bi bio ispunjen svijetlo plavom bojom).



Slika 6.3 – Prikaz izgleda prozora aplikacije Normal Maker [25]

Pritiskom na gumb označen tekstom „Load Texture“ otvara se prozor koji prikazuje direktorije sadržaja računala, te prikazuje prisutne .png, .jpg i .tif datoteke. Izborom neke datoteke odgovarajućeg formata i pritiskom na

gumb „Open“ stvara se kopija teksture u radnoj memoriji, te se ona prikazuje na ekranu.

Nakon primjene željenih efekata na radnu teksturu, pritiskom na „Save Screen“ otvara se prozor koji prikazuje direktorije sadržaja računala. Nakon odabira željenog direktorija, imenovanja nove slikovne datoteke i pritiskom na „Save“, tekstura koja je trenutno prikazana u aplikaciji se pohranjuje u novu teksturu u radnoj memoriji.

Između izbora efekata te promjene parametara efekata, učitavanja i pohrane teksture, u aplikaciji su prisutna tri gumba sa sljedećim funkcijama: gumbi s tekstovima „Show Original“ i „Show Working“ pritiskom reagiraju tako da isključuju onog drugog i ostaju pritisnuti dok ih se ne isključi, a kao efekt imaju prikaz originalne odnosno radne teksture, za razliku od teksture trenutnog prikaza, koja se prikazuje ukoliko niti jedna od tih opcija nije uključena. Treći gumb označen tekstom „Restore Original“ obnavlja vrijednosti radne teksture na vrijednosti originalno učitane teksture.

7. Zaključak

Teksture u računalnoj grafici daju razinu realizma nevjerojatnim pričama i nestvarnim svjetovima, prikazuju izgled i predočuju karakter virtualnih likova, te stvaraju dojam druge stvarnosti u kojoj je sve moguće.

Izrada tekstura je uglavnom problem umjetničke prirode, te zahtjeva prirodan talent i nadahnuće. U ovom radu je obrađen malen dio tekstura posebnih primjena, koje na oslikanu razinu umjetnosti dodaju realističnost prikaza uz nizak broj korištenih grafičkih primitiva, kako bi cijela scena mogla imati što više vizualnih efekata koji pobuđuju maštu i radost. No za razliku od umjetničkog pristupa izradi tekstura difuzne komponente, teksture opisane u ovom radu zahtjevaju preciznost, tehničko razumijevanje značenja boja teksture te okolnosti primjene tekstura za dodatne vizualne detalje.

Glavni zadatak ovog rada bio je predstaviti osnove teorije mapa, načine izrade iz različitih ulaznih podataka i primjene mapa u različite svrhe, te usporediti kvalitete rezultata primjene.

Rezultati koji su dobiveni programskom implementacijom su zadovoljavajući s obzirom na zamišljenu ideju korištenja aplikacije, no postoji još puno prostora za proširenje metoda i kvalitete izrade, kao i integriranju više tehnika za izradu mapa iz svih spomenutih ulaznih podataka.

Što se tiče primjena, neke su dovedene do granice za trenutnu tehnologiju i formate zapisa tekstura, a neke metode se i dalje oslanjaju na najviše zadovoljavajuće aproksimacije, te čekaju razvoj boljih algoritama i računalnih komponenti.

Računalna grafika je moderno platno izražavanja velikog broja vrhunskih umjetnika, neiscrpan izvor priča i zabave, te sredstvo za prikazivanje nezamislivog, nepostojećeg i još neispričanog, koje nikad ne bi bilo isto bez posebnih tehnika za dodavanje najsitnijih detalja s ove strane stvarnosti.

8. Literatura

[1] Teorija mapa normala i primjena u praksi – pristup stranici u periodu 24. 4. 2012. – 3. 6. 2012. http://wiki.polycount.com/NormalMap#Anti-Aliasing

[2] Teorija mapa normala - pristup stranici u periodu 4. 5. 2012. – 27. 5. 2012. http://www.svartberg.com/tutorials/article_normalmaps/normalmaps.html

[3] Teorija mapa normala - pristup stranici u periodu 2. 5. 2012. – 29. 5. 2012. http://www.3dkingdoms.com/tutorial.htm

[4] Program preuzet 12. 4. 2012. Sa stranice http://www.adobe.com/cfusion/tdrc/index.cfm?product=designweb_premium&loc=en_us

[5] Program preuzet 26. 4. 2012. sa stranice http://www.luxology.com/trymodo/

[6] Program preuzet 29. 4. 2012. sa stranice http://www.crazybump.com/

[7] Slika preuzeta 25. 5. 2012. sa stranice http://earthexplorer.usgs.gov/metadata/3191/SRTM1N47W105/, nastalo 2. 2002.

[8] Slika preuzeta 12. 5. 2012. sa http://newauthors.wordpress.com/2010/07/06/cars-brick-walls-and-learning-to-drive/

[9] Slika preuzeta 12. 5. 2012. sa http://basictextures.com/free-textures/stone-brick-wall-red-scratches-00353.html

[10] Slika preuzeta 12. 5. 2012. sa http://www.officialpsds.com/-----Brick-Wall-With-Lights--stock4221.html

[11] Slika preuzeta 12. 5. 2012. sa http://sv.wikipedia.org/wiki/Fil:Soderledskyrkan_brick_wall.jpg, 13. 4. 2006.

[12] Slika preuzeta 12. 5. 2012. sa http://www.kaneva.com/mykaneva/PictureDetail.aspx?assetId=5931023, Solna_Karolinska_institutet_Brick_wall02, autor: blueshift2, 26. 10. 2009.

[13] Slike preuzete 12. 5. 2012. iz http://sirkan.iit.bme.hu/~szirmay/egdisfinal3.pdf

[14] Slika preuzeta 18. 5. 2012. sa http://www.chrisalbeluhn.com/Normal_Map_Tutorial.html

[15] Slika preuzeta 16. 5. 2012. sa http://www.piculous.com/wp-content/uploads/2008/11/terragen___the_way_god_made_me.jpg

[16] Slika preuzeta 16. 5. 2012. sa http://inmyotherworld.free.fr/gfx/terragen/01.jpg

[17] Slika preuzeta 16. 5. 2012. sa http://store.2753productions.com/images/uploads/Twisted%20Desert%20Small.jpg

[18] Slika preuzeta 16. 5. 2012. sa http://www.1stwebdesigner.com/inspiration/stunning-digital-nature-artworks-terragen/

[19] Slike preuzete 19. 5. 2012. sa http://wiki.polycount.com/NormalMap#Anti-Aliasing

[20] Slike preuzete 20. 5. 2012. sa http://zarria.net/nrmphoto/nrmphoto.html

[21] Slika preuzeta 22. 5. 2012. sa http://webtreats.mysitemyway.com/tileable-basket-weave-textures/

[22] Slika preuzets 24. 4. 2012. sa http://wiki.polycount.com/NormalMap#Anti-Aliasing

[23] Slika preuzeta 17. 5. 2012. sa http://www.chrisalbeluhn.com/Normal_Map_Tutorial.html

[24] Slika preuzeta 18. 5. 2012. sa http://en.wikipedia.org/wiki/File:Bikesgray.jpg, autor Davidwkennedy, 30. 7. 2007.

[25] Slika na ekranu aplikacije preuzeta 22. 5. 2012. od Ane Stepić, nepoznatog autora

[26] Osnove dobivanja mapa normala od modela visoke i niske rezolucije, pristup stranici u periodu 3. 4. 2012. – 24. 5. 2012. http://www.game-artist.net/forums/spotlight-articles/43-tutorial-introduction-normal-mapping.html

[27] Dobivanje mapa normala pomoću fotografija, pristup stranici u periodu 7. 5. 2012. – 22. 5. 2012. http://zarria.net/nrmphoto/nrmphoto.html

[28] Teorija primjene visinskih mapa, pristup stranici u periodu 8. 5. 2012. – 3. 6. 2012. http://sirkan.iit.bme.hu/~szirmay/egdisfinal3.pdf

[29] Perlinov šum, pristup stranici u periodu 2. 5. 2012. – 21. 5. 2012.
http://freespace.virgin.net/hugo.elias/models/m_perlin.htm

[30] Algoritam dijamant – kvadrat, pristup stranici u periodu 16. 5. 2012. – 26. 5. 2012. http://www.gameprogrammer.com/fractal.html#midpoint

[31] Perlinov šum, pristup stranici u periodu 2. 5. 2012. – 21. 5. 2012. http://webstaff.itn.liu.se/~stegu/TNM022-2005/perlinnoiselinks/perlin-noise-math-faq.html

[32] Vektori u Phongovom modelu sjenčanja, slika preuzeta 2. 6. 2012. sa stranice http://en.wikipedia.org/wiki/File:Blinn_Vectors.svg, autor Martin Kraus, 17. 7. 2011

[33] Tangencijalni prostor i dobivanje TBN matrice – pristup stranici 5. 6. 2012., http://www.blacksmith-studios.dk/projects/downloads/tangent_matrix_derivation.php , Blacksmith Studios, autor Jakob Gath, 2006.

9. Sažetak

Postupci izrade i primjene mapa normala

Ovaj rad opisuje osnovne koncepte izrade i primjene posebnih tekstura u računalnoj grafici: mapa normala i visinskih mapa. Rad je podijeljen u pet poglavlja. Prvo poglavlje daje uvod u prikaz modela objekta kao skupa podataka, pokriva osnovne vrste podataka modela i njihovu svrhu u primjeni, te sadrži teoriju tangencijalnog prostora i postupak izvoda matrice tangencijalnog prostora. Drugo poglavlje posvećeno je detaljnom opisu teorije visinskih mapa, te detaljima u različitim načinima izrade i primjene. U trećem poglavlju dan je opis konkretne programske implementacije aplikacije za izradu visinskih mapa. Četvrto poglavlje pokriva uvod u teoriju mapa normala, osnovne načine izrade i primjene, te usporedbu rezultata i tehnika s visinskim mapama. Peto poglavlje sadrži opis logike i implementacije aplikacije za izradu i manipulaciju mapa normala.

Ključne riječi: Visinske mape, mape normala, preslikavanje neravnina

Abstract

Methods of normal map synthesis and practical application

This paper describes the basic concepts of synthesis and application of specific textures in computer graphics: normal maps and height maps. It is divided into five chapters. The first chapter provides an introduction to the presentation of the model object as a set of data, covers the basic types of data and their purpose in practical appliance, and it contains a theoretical introduction in tangent space, together with the process of deriving tangent space transformation matrix. The second chapter is devoted to a detailed description of the theory of height maps, and details in various methods of height map synthesis and implementation. The third chapter contains a description and implementation details of application software able to create height maps. The fourth section covers an introduction to the theory of normal maps, basic methods of synthesis and appliance, and comparison of results and mentioned techniques. The fifth chapter contains a description of

the implementation for created application able to create and manipulate normal maps.

Keywords: height maps, normal maps, bump mapping, normal mapping