

Virtualna okruženja

Laboratorijske vježbe

Vježba 1

Programiranje grafičkog sklopovlja

FER – ZTEL – Tomislav Pejša, Igor S. Pandžić, Sara Vlahović

Suradnja na pripremi vježbe:

Tomislav Čoh

Jakov Dužević

Zoran Hranj

Filip Kiš

Josip Iveković

Uvod

Razni prirodni i fizički zakoni zajedničkim djelovanjem čine bogati svijet i okolinu svuda oko nas. Poznavajući te prirodne pojave i sa sve većim napretkom u kompjutorskoj tehnologiji moguće ih je simulirati na računalima tako da je gotovo nemoguće primijetiti razliku između stvarne i simulirane scene. Međutim, potrebni izračuni su obično prekompleksni da bi bili izvedivi u stvarnom vremenu. Stoga se za simuliranje tih složenih pojava u stvarnom vremenu koriste aproksimativne tehnike koje nazivamo specijalni efekti. Među specijalne efekte ubrajaju se simulacije svjetlosti, sjene, zrcaljenja, čestica, itd. Cilj ove vježbe je upoznavanje s implementacijom specijalnih efekata korištenjem jezika za sjenčanje visoke razine.

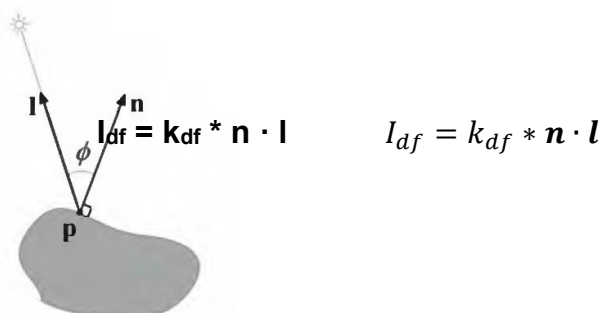
1 Potrebni alati

Za izvođenje vježbe potrebni su materijali dostupni na stranicama predmeta te računalo s Windows operacijskim sustavom i instaliranom najnovijom verzijom Microsoft DirectX Runtimea. Na vlastitom računalu potrebno je otpakirati materijale dostupne na stranicama predmeta, pokrenuti izvršnu datoteku i u omiljenom tekstualnom editoru modificirati datoteke programa za sjenčanje. Ako netko želi modificirati postojeći C++ projekt, mora imati instaliran Visual Studio 2010 te DirectX SDK. Studenti koji iz bilo kojeg razloga ne mogu obaviti laboratorijsku vježbu na vlastitom računalu mogu se pravovremeno obratiti na mail vo@fer.hr kako bi im se omogućilo korištenje zavodskih računala.

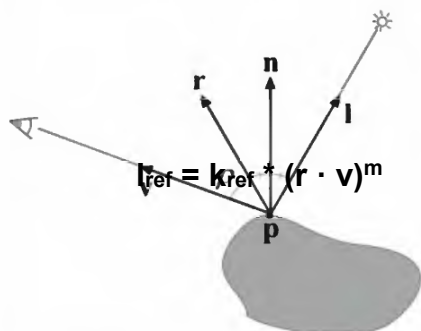
2 Teorijska podloga

2.1 Model osvjetljenja

Osvjetljenje označava interakciju između izvora svjetlosti te materijala i geometrije objekta na promatranoj sceni, tj. na sceni koja se iscrtava. U interaktivnoj 3D grafici najčešće se koristi Phongov model osvjetljenja, kod kojeg se osvjetljenje sastoji od tri osnovne komponente: difuzne, reflektirajuće (*specular*) i ambijentalne. Difuzna komponenta ima najviše veze s fizičkom realnošću interakcije svjetla i površine. Ona se temelji na Lambertovom zakonu koji kaže da je reflektirana svjetlost savršeno difuzne površine (mat, bez sjaja) određena kosinusom između normale površine i vektora smjera izvora svjetlosti. Računa se pomoću formule:

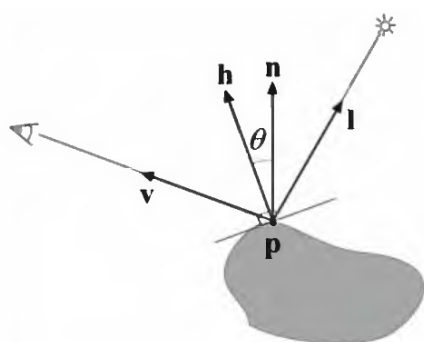


Za razliku od difuzne komponente, koja ne ovisi o smjeru gledanja (tj. poziciji kamere u našem slučaju), reflektirajuća ovisi. Ova komponenta naglašava zakrivljenja objekta, te pomaže gledatelju određivanje smjera i intenziteta izvora svjetlosti. Formula za izračun ove komponente glasi:



$$I_{ref} = k_{ref} * (r \cdot v)^m$$

gdje je k_{ref} koeficijent utjecaja reflektirajuće komponente (ovisi o materijalu), a m koeficijent jačine reflektiranja (također ovisi o materijalu), r vektor smjera izvora svjetlosti zrcaljen oko normale, a v vektor smjera gledanja. Međutim, kako je izračun zrcalne komponente smjera gledanja oko normale skupo računati, može se koristiti i pojednostavljena formula:



$$I_{ref} = k_{ref} * (n \cdot h)^m \quad I_{ref} = k_{ref} * (n \cdot h)^m$$

gdje je n vektor normale, a h takozvani polu-vektor između l i v :

$$h = \frac{v + l}{\|v + l\|}$$

Do sada smo uzimali u obzir svjetlost koja dolazi na objekt direktno iz izvora. Međutim, u stvarnom svijetu svjetlost se reflektira od drugih objekata (npr. od zida). Kako bi se simulirao taj efekt, dodaje se ambijentalno svjetlo koje je uglavnom konstanta za pojedinu scenu. Stoga je formula ambijentalne komponente jednostavna:

$$I_{amb} = k_{amb} I_{amb} = k_{amb}$$

Na taj način i strane objekta koje su okrenute suprotno od izvora svjetlosti dobiju određeno osvjetljenje te nisu potpuno tamne. Ukupni efekt osvjetljenja koji se računa u pojedinoj točki je stoga:

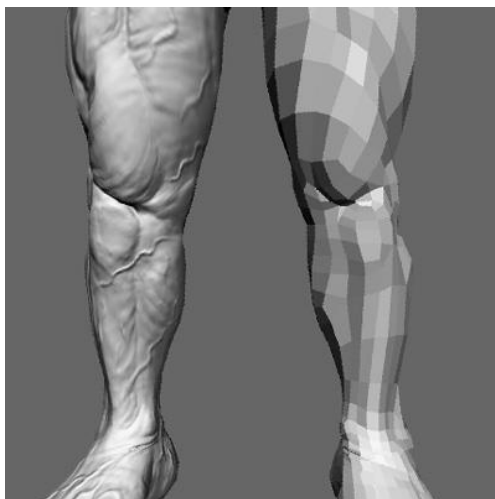
$$I_{uk} = I_{amb} + I_{dif} + I_{ref} \quad I_{uk} = I_{amb} + I_{dif} + I_{ref}$$

Sjenčanje je proces računanja osvjetljenja i određivanja boje pojedinog pixela. Postoje 3 osnovna modela sjenčanja: plošno, Gouraudovo, Phongovo. Plošno računa osvjetljenje za svaki poligon, te svi pixeli istog poligona imaju isto osvjetljenje. Gouraudovo računa osvjetljenje za vrhove poligona te zatim izračunate vrijednosti interpolira kroz poligon.

Phongovo sjenčanje interpolira normale vrhova kroz poligon te u svakom pixelu izračunava osvjetljenje posebno. Zbog toga je Phongovo sjenčanje najskuplje, ali istovremeno daje najbolje rezultate.

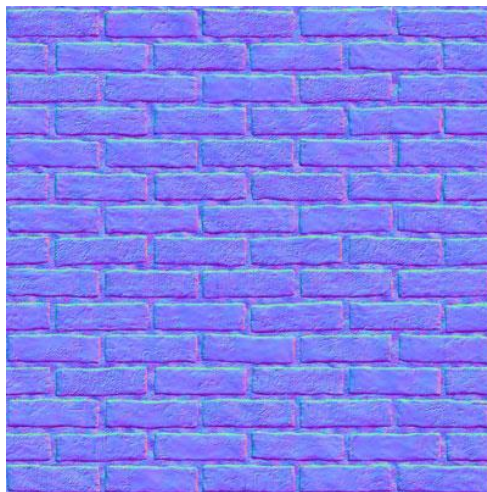
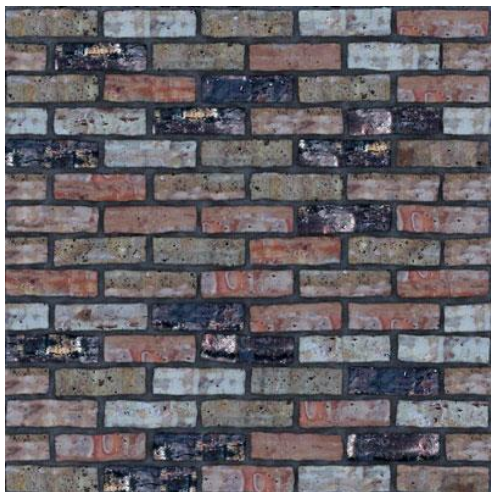
2.2 Preslikavanje neravnina

Preslikavanje neravnina je metoda koja omogućava prikazivanje reljefnih i neravnih površina bez potrebe stvaranja dodatne geometrije. Kako smo mogli vidjeti do sada, za računanje osvjetljenja potrebne su normale pojedinih točaka modela. Osnovna ideja ove metode je da se objekt modelira sa što manjim brojem poligona, a da se stvarne normale pojedine točke spremne u teksturu. Zatim se kod iscrtavanja, umjesto normala poligona, koriste normale iz teksture, čime se dobije efekt reljefnosti, a očuva brzina iscrtavanja zbog znatno smanjenog broja poligona.



Slika 1. Desno je prikazana poligonalna geometrija noge, dok je lijevo prikazana noga sa identičnom geometrijom, uz korištenje preslikavanja neravnina.

Normala je u teksturi spremljena tako da svaka od tri boje slike (RGB) predstavlja jednu koordinatu (x, y, z), i to tako da 0 vrijednost boje predstavlja -1 vrijednost koordinate, a 255 vrijednost 1.



Slika 2. Primjeri obične teksture (lijevo) i njene odgovarajuće teksture normala (desno)

Normale u teksturi normala su zadane u koordinatnom sustavu tangente koji je određen vektorima tangente, normale i binormale i jedinstven za svaki poligon. Prednost takve teksture normala je u tome što se može koristiti na proizvoljnoj površini bez potrebe za ponovnim proračunom normala.

U programu za sjenčanje fragmenata je prije proračuna osvjetljenja potrebno normale transformirati u koordinatni sustav scene ili kamere. Alternativni pristup je da se u programu za sjenčanje vrhova vektor svjetlosti i pogleda transformiraju u sustav tangente te da se proračuni osvjetljenja rade u tom sustavu. Time se poboljšavaju performanse, jer nema potrebe za računanjem transformacija za svaki fragment posebno.

2.3 Preslikavanje sjena

Sjene su važan element u pokušaju prikazivanja što realističnije scene jer daju gledatelju osjećaj odnosa i položaja objekata. Jedna od tehnika za postizanje efekta sjena u stvarnom vremenu je preslikavanje sjena. Ideja tehnike je da se scena prvo iscrtava iz perspektive svakog izvora svjetlosti, za što je potrebno onoliko prolaza koliko ima izvora, pa tek onda iz perspektive kamere. Prilikom iscrtavanja iz perspektive svjetla koristi se samo Z-spremnik koji pohranjuje dubinu, tj. udaljenosti točaka od tog izvora svjetlosti. Podaci iz Z-spremnika spremaju se u zasebne teksture koje nazivamo teksturama sjena (engl. *shadow maps*). U posljednjem prolazu scena se iscrtava iz perspektive kamere, a teksture sjena koriste se da bi se za svaku pojedinu točku odredilo nalazi li se u sjeni ili ne. Ovaj prolaz je implementiran u programima za sjenčanje.

Ovakav način iscrtavanja sjena uvelike usporava ukupnu brzinu iscrtavanja jedne slike (engl. *frame*). Zbog toga se za prolaze koji iscrtavaju scenu iz perspektive izvora ne koriste boje, koristi se plošni model sjenčanja te uklanjanje prednjih poligona. Prednji poligoni se uklanjaju kako bi se barem djelomično riješio problem samosjenčanja (engl. *selfshadowing*). Za dodatno ubrzanje može se koristiti iscrtavanje u pomoćni spremnik (engl. *off-screen rendering*) tj. umjesto u standardni spremnik slike (engl. *frame buffer*) čiji se sadržaj prikazuje na ekranu crta se u pomoćni spremnik čiji se sadržaj može direktno upisivati u teksturu. Ova tehnika se ne koristi u vježbi jer je u potpunosti podržana tek u grafičkim procesorima koji podržavaju *Shader Model 3.0*.

U programu za sjenčanje potrebno je položaj vrha transformirati u koordinatni sustav svake pojedine teksture sjena, i to korištenjem sljedeće formule:

$$[s, t, r, q]^T = B * P_L * V_L * M_C * [x, y, z, w]^T \quad [s, t, r, q]^T = B * P_L * V_L * M_C * [x, y, z, w]^T \quad (1)$$

Redoslijed primjene matrica je s desna na lijevo: prvo množimo vrh s transformacijskom matricom objekta (engl. *model matrix*) kako bismo vrh prebacili u koordinatni sustav scene te transformacijskom matricom pogleda (engl. *view matrix*) izvora svjetlosti. Zatim množimo standardnom matricom perspektivne projekcije koja transformira vrh u jediničnu kocku s koordinatama u rasponu $[-1, 1]$. Konačno, matricom korekcije B (engl. *bias*) vrši se pomak koordinata iz intervala $[-1, 1]$ u interval $[0, 1]$ što odgovara u, v koordinatama teksture.

Virtualna okruženja – laboratorijske vježbe FER-ZTEL, Tomislav Pejša, Igor S. Pandžić, Sara Vlahović

Nakon transformacije koordinata radi se uzorkovanje teksture sjena tj. dohvaćanje boje teksture (koju interpretiramo kao dubinu) na izračunatim koordinatama i uspoređivanje s dubinom u trenutnoj točki. Ako je trenutna dubina veća, točka se nalazi u sjeni i potrebno ju je na odgovarajući način iscrtati, npr. korištenjem samo ambijentalne komponente svjetlosti, skaliranjem difuzne komponente svjetlosti, skaliranjem ukupne boje fragmenta itd.

3 Opis zadatka

Potrebno je nadopuniti datoteke programa za sjenčanje kako bi uspješno simulirali efekte preslikavanja neravnina i preslikavanja sjena.

4 Upute za rad

4.1 Preslikavanje neravnina

Pokretanjem *NormalMapping.exe* izvršne datoteke koja se nalazi u direktoriju *VO-VX-SpecijalniEfekti\Neravnine* iscrtava se scena koja sadrži kocku na čije stranice je mapirana određena tekstura.

Pomoću tekstualnog editora (npr. *Notepad*) potrebno je otvoriti datoteku *NormalMapping.fx* (koja se nalazi u istom direktoriju) i u njoj pronaći definiciju funkcije procesora točaka (*pixel shader*). Ta funkcija trenutno implementira samo osnovni model osvjetljenja (koristeći interpolirane normale vrhova, dakle Phongovo sjenčanje, i samo ambijentalno i difuzno svjetlo). Funkciju je potrebno nadopuniti tako da umjesto trenutne normale pročita normalu iz teksture. Osim toga potrebno je, koristeći vektor izvora svjetla i vektor smjera gledanja izračunati i reflektirajuću komponentu i dodati je na ukupni doprinos svjetla.

Funkcija *tex2D(_sampler_texture, _koordinata_texture_)* učitava vrijednosti teksture na određenim koordinatama. Kako su boje u HLSL jeziku predstavljene u intervalu $[0,1]$, da bi se iz teksture izvukle normale potrebno ih je skalirati na interval $[-1,1]$.

Funkcija *pow(varijabla,potencija)* potencira varijablu koja može biti skalar, vektor ili matrica.

Po završetku izmjena potrebno je spremati promjene i ponovo pokrenuti izvršnu datoteku (*NormalMapping.fx* datoteka ne smije promijeniti ime), koja bi sada trebala prikazivati izmijenjenu scenu (tj. neravnine na kocki).

Potrebno je napraviti slike početne i izmijenjene scene (po mogućnosti rotirati kocku da se bolje vide efekti), te ih priložiti u izvještaj.

4.2 Preslikavanje sjena

Pokretanjem *ShadowMapping.exe* izvršne datoteke koja se nalazi u direktoriju *VO-VX-SpecijalniEfekti\Sjene* iscrtava se scena s logom Fakulteta u prostori sa jednim izvorom svjetla.

Pomoću tekstualnog editora (npr. *Notepad*) potrebno je otvoriti datoteku *ShadowMapping.fx* (koja se nalazi u istom direktoriju) i u njoj pronaći definiciju funkcije procesora točaka (*pixel shader*) za scenu (postoji i procesor točaka za sjene). Ta funkcija trenutno osvjetljava sve točke u dometu reflektora (ne uzimajući u obzir sjene). Funkciju je potrebno nadopuniti tako da pročita udaljenosti zapisane u x komponentu teksture (*shadowMap*). To se postiže ranije opisanom funkcijom

`tex2D(_sampler_texture, koordinate_texture_)`. Međutim, kako smo teksturu kreirali iz perspektive svjetla, koordinata točke iz pogleda kamere ne odgovara koordinati teksture pa je potrebno koristiti koordinate dobivene transformacijom u prostor svjetla. Tako transformirane koordinate je potrebno svesti na upotrebljiv interval vrijednosti $[0,1]$ u koordinatnom sustavu teksture. Nakon što se učitaju spremljene udaljenosti, potrebno ih je usporediti sa udaljenošću trenutne točke. Da bi se to postiglo, potrebno je koristiti sljedeću prilagođenu formulu:

$$ucitana_udaljenost \geq (IN.viewPosLight.z/xMaxDepth - 0.01)$$

Faktor 0.01 je dodan kako bi se izbjegla pojava samosjenčanja uslijed pogreške koja nastaje zbog nepreciznosti pohranjivanja rezultata dijeljenja u varijablu tipa *float*.

Ako je navedena usporedba točna, znači da se točka nalazi na svjetlu te je treba u potpunosti osvijetliti. U protivnom se točka nalazi u sjeni te ju je potrebno osvijetliti samo ambijentalnim svjetlom.

Po završetku editiranja potrebno je spremiti promjene i ponovo pokrenuti izvršnu datoteku (*ShadowMapping.fx* datoteka ne smije promijeniti ime), koja bi sada trebala prikazivati scenu s uključenim sjenama.

Potrebno je napraviti slike početne i izmijenjene scene, te ih priložiti u izvještaj.

Primijetite kvalitetu rubova sjene te obrazložite zašto dolazi do te pojave.

5 Predavanje rezultata vježbe

Rezultati vježbe predaju se korištenjem sustava *Moodle* u obliku **PDF** datoteke imena **VO-V1-<ImePrezime>**, koja treba sadržavati:

- Izvještaj o izvođenju vježbe – kratak opis postupka izrade cijele vježbe sa slikom predmeta/scene prije i poslije primjene navedenih efekata
- Izmijenjene isječke koda (uz objašnjenje)

Napomena: Rezultati se šalju isključivo preko gore navedene aplikacije. U slučaju problema, javiti se e-mailom na adresu vo@fer.hr. Sačuvajte kopiju poslanih rezultata.

6 Napredni zadaci (ovaj dio nije obavezan)

Izradite trodimenzionalni model kocke i razvucite njene stranice kako biste dobili model nalik zidu. Odaberite i preuzmite *stock* fotografiju teksture popločane ceste ili zida od opeka te je iskoristite kao teksturu modela. Potom upotrijebite odabranu fotografiju kako biste izradili mapu normala. Usporedite izgled teksturiranog zida sa i bez korištenja mape normala. Za izradu mape i modela možete koristiti Gimp, Photoshop, Blender, Unity ili druge alate po izboru. U izvještaj dodajte opis izrade naprednog zadatka te odgovarajuće *screenshotove*.

6 Literatura

- [1] Randi J. Rost. *OpenGL Shading Language, Second Edition*. Addison Wesley Professional. 2006.
- [2] John Kessenich, Dave Baldwin, Randi J. Rost. *The OpenGL Shading Language*. Language version: 1.20. Document Revision: 8.
<http://www.opengl.org/documentation/glsl/>
- [3] Lengyel, Eric. *Computing Tangent Space Basis Vectors for an Arbitrary Mesh*. Terathon Software 3D Graphics Library, 2001.
<http://www.terathon.com/code/tangent.html>
- [4] Mark J. Kilgard. *A practical and Robust Bump-mapping Technique for Today's GPUs*. NVIDIA Corporation. 2000.
http://developer.nvidia.com/object/Practical_Bumpmapping_Tech.html
- [5] Jérôme Guinot. *Lighting with GLSL Phong model*. 2006.
http://www.ozone3d.net/tutorials/glsl_lighting_phong.php