FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ZAVOD ZA TELEKOMUNIKACIJE

1. Laboratorijska vježba iz kolegija "Virtualna okruženja"

Nenad Mikša

Zagreb, 2011.

Sadržaj

Uvod

U prvoj laboratorijskoj vježbi iz kolegija "Virtualna okruženja" se je obrađivala tema specijalnih efekata. Konkretno, bilo je potrebno doraditi postojeće programe za sjenčanje kako bi isti računali spekularnu komponentu svake točke, osim difuzne i ambijentalne koje su već bile isprogramirane. Nakon toga je bilo potrebno dodati podršku za preslikavanje neravnina i sjena kako bi se iscrtana scena doimala puno realističnijom.

Tijekom izrade ove laboratorijske vježbe korištena je verzija vježbe bazirana na Java+OpenGL platformi pod operacijskim sustavom Linux. Korištena grafička kartica je NVIDIA GeForce GTX 470 sa verzijom pogonskog programa (*drivera*) 285.05.09 koji omogućuje verziju OpenGL-a i GLSL-a 4.2 (*Shader Model 5.0*).

Zadatak 1: Računanje spekularne komponente osvijetljenja

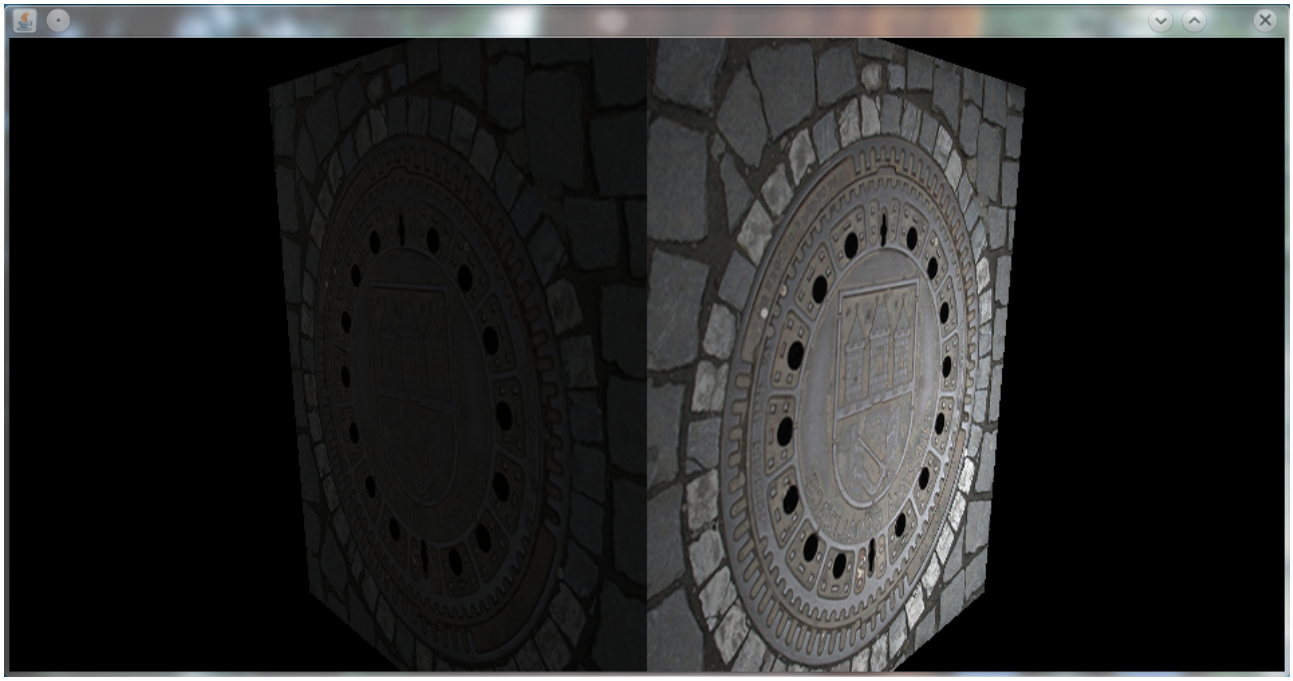
Funkcija *spotLight* programa za sjenčanje točaka proračunava ambijentalnu, spekularnu i difuznu komponentu osvijetljenja svake točke. Proračun ambijentalne i difuzne komponente je već implementiran, a u zadatku je potrebno implementirati i proračun spekularne komponente. Konkretno, potrebno je odrediti vrijednost parametra *specFlare* koji predstavlja koeficijent kojim se množi svojstvo materijala u konkretnoj točki za dobivanje konačne boje točke. Navedeni parametar ovisi o kosinusu kuta između vektora pogleda i vektora reflektiranog svjetla.

```
vec3 reflektiraniVektorSvjetla = -reflect(lightDir, normal);
// kosinus kuta dva normalizirana vektora je dot produkt
float kosinusKuta = dot(viewDir, reflektiraniVektorSvjetla);
specFlare = gl_FrontMaterial.specular * pow(max(kosinusKuta,
0.0), gl_FrontMaterial.shininess);
```

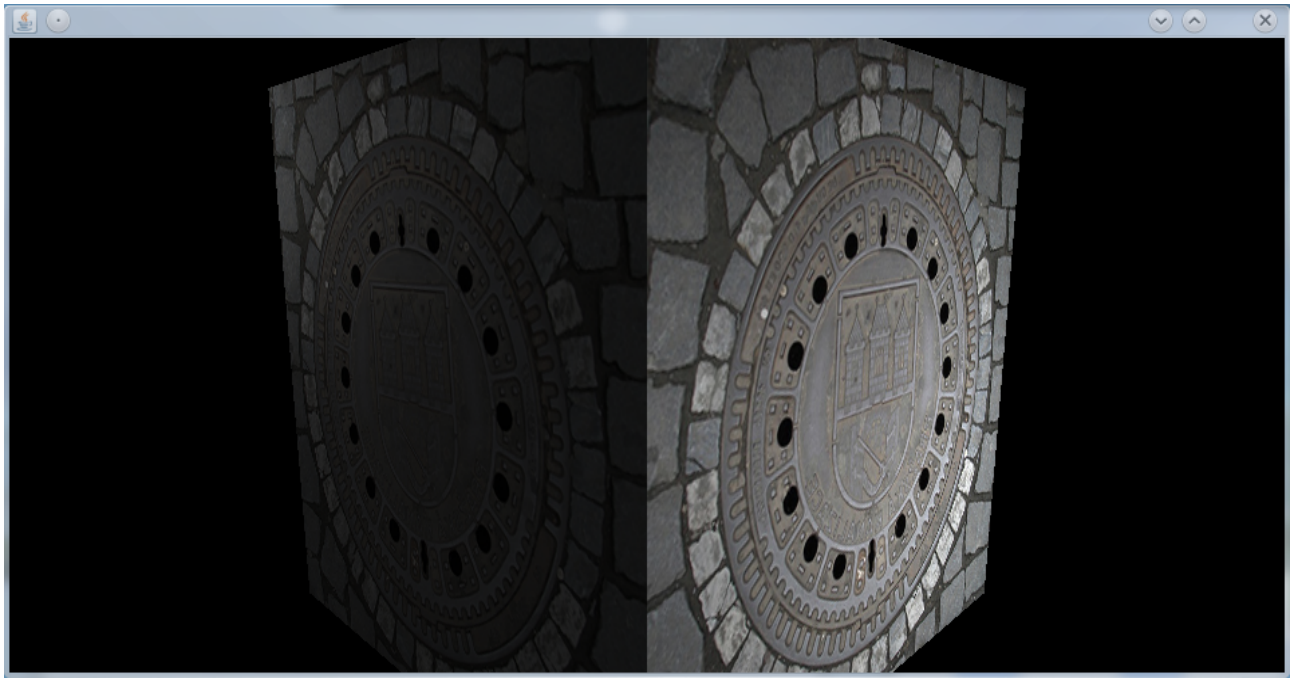
Ispis 1: Proračun parametra specFlare

Dio programa za sjenčanje točaka koji radi proračun parametra *specFlare* je prikazan u ispisu 1. Program prvo izračuna vektor smjera reflektirane zrake svjetla pomoću naredbe *reflect*, zatim izračuna kosinus kuta između vektora smjera reflektirane zrake i vektora smjera kamera odnosno pogleda na način da izračuna njihov skalarni umnožak. Budući da su i vektor *viewDir* i vektor *reflektiraniVektorSvjetla* oba jedinični, njihov skalarni umnožak je jednak kosinusu kuta kojeg oni međusobno zatvaraju. Konačna vrijednost parametra se računa kao spekularni koeficijent materijala trenutne točke pomnožen sa kosinusom kuta potenciranim na koeficijent sjajnosti materijala točke. Iako se parametar može izračunati i na drugačije načine (bitno je da ovisi o kosinusu kuta između vektora pogleda i vektora reflektirane zrake svjetla), empirijski je utvrđeno da upravo navedena formula daje oku najugodniji rezultat.

Prikaz prve scene (kocke) prije računanja spekularne komponente svake točke je vidljiv na slici 1, a nakon dodavanja programskog odsječka za računanje spekularne komponente na slici 2. Moguće je uočiti malo vjerniji prikaz osvijetljene kocke.



Slika 1: Prikaz kocke prije preslikavanja neravnina i računanja spekularne komponente svjetla svake točke



Slika 2: Prikaz kocke prije preslikavanja neravnina, ali nakon računanja spekularne komponente svjetla svake točke

Zadatak 2: Preslikavanje neravnina

U ovom zadatku je bilo potrebno doraditi program za sjenčanje točaka kako bi isti omogućio prikaz neravnina kocke. Iluzija neravnine se dobiva mijenjanjem normala svake točke u skladu sa teksturom normala. Budući da su normale u teksturi normale zadane u koordinatnom sustavu tangente, prije svih ostalih operacija je potrebno sve vektore koje koristimo transformirati u isti koordinatni sustav. To činimo tako da vektore smjera svjetla i pogleda pomnožimo sa matricom

transformacije u koordinatni sustav tangente koja je izračunata za svaki vrh u programu za sjenčanje vrhova.

```
if (BumpMapping) {
      lightDir = lightDir * TangentMatrix;
      spotDir = spotDir * TangentMatrix;
      viewDir = viewDir * TangentMatrix;
}
```

Ispis 2: Transformacija vektora svjetlosti i pogleda u prostor tangente

Programski odsječak koji provodi navedenu transformaciju je prikazan u ispisu 2.

Osim navedenog, potrebno je i učitati normalu iz teksture normala. U suprotnom bi sve normale svih točaka bile iste te bi površina bila glatka. Programski odsječak koji učitava normalu iz teksture normala je prikazan u ispisu 3.

```
if(BumpMapping) {
     vec4 texNormala = texture2D(NormalTexture, gl_TexCoord[0].xy);
     normal = normalize(texNormala.xyz);
} else {
     normal = normalize(Normal);
}
```

Ispis 3: Učitavanje normale iz teksture normala

Kao što je vidljivo u ispisu 3, normalu točke učitavamo iz teksture normala samo ako želimo koristiti preslikavanje neravnina (zastavica *BumpMapping* je istinita). U suprotnom koristimo običnu normalu. Rezultat preslikavanja neravnina je prikazan na slici 3.

Slika 3: Efekt preslikavanja neravnina

Zadatak 3: Preslikavanje sjena

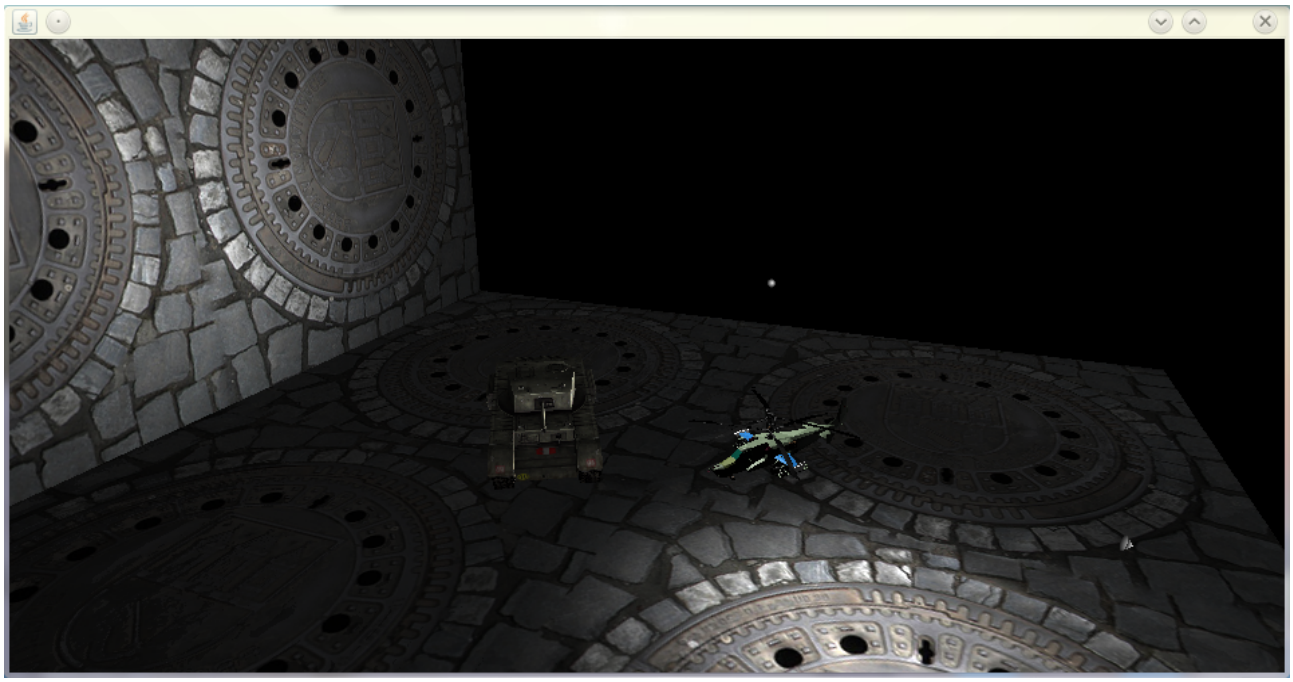
U zadnjem dijelu vježbe je bilo potrebno doraditi program za sjenčanje točaka kako se iscrtavale sjene tenka i helikoptera u sceni. Programski odsječak koji to radi je prikazan u ispisu 4.

```
if(ShadowMapping){
      vec4 pozicijaTocke = shadowMatrix * vec4(Position, 1.0);
      float pikselJeUSjeni=shadow2DProj(ShadowTexture, pozicijaTocke).r;
      if(pikselJeUSjeni==0.0) {
           spot*=0.25;
      }
}
```

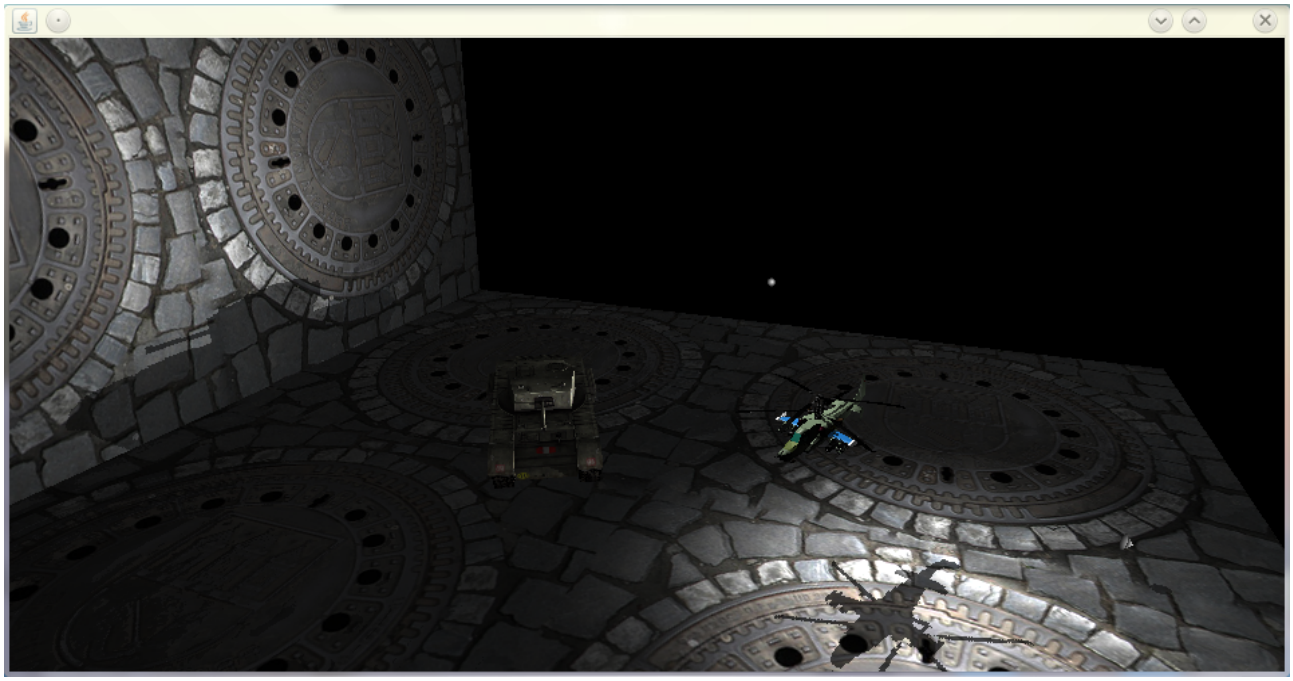
Ispis 4: Programski odsječak za preslikavanje sjena

Programski odsječak se izvodi samo ako je uključeno preslikavanje sjena (zastavica *ShadowMapping* je istinita). Preslikavanje sjena se radi na način da se trenutna pozicija točke napiše pomoću homogenih koordinata te se tako zapisana transformira u koordinatni sustav teksture sjene. Tako transformirane koordinate se zatim predaju funkciji *shadow2DProj* koja iz zadane teksture sjene i pozicije točke odredi nalazi li se točka unutar sjene ili ne. Ako se točka nalazi unutar sjene, intenzitet svjetla za tu točku smanjujemo na 25% prvotne vrijednosti.

Scena tenka i helikoptera bez preslikavanja sjena je prikazana na slici 4, a ista scena s uključenim preslikavanjem sjena na slici 5.



Slika 4: Scena bez preslikavanja sjena



Slika 5: Scena s preslikavanjem sjena