SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

2020

Virtualna okruženja

Laboratorijske vježbe

Vježba 1

Programiranje grafičkog sklopovlja

Jakov Dorontić, 0036503608

1. Uvod

U ovoj laboratorijskoj vježbi zadatak je bilo implementirati dvije vrste specijalnih efekata na grafičkom sklopovlju pomoću jezika HLSL (*eng. High-Level Shader Language*) koji je razvijen od strane Microsoft korporacije za Direct3D 9 API. Specijalni efekti (aproksimativne tehnike) koje su implementirane u ovoj laboratorijskoj vježbi su preslikavanje neravnina i preslikavanje sjena.

2. Preslikavanje neravnina

Preslikavanje neravnina (*eng. normal mapping*) je tehnika prikaza predmeta kod koje se za svaki slikovni element površine objekta provjerava i računa odstupanje od uobičajene boje elementa s ciljem postizanja prividnog povećanja kompleksnosti geometrije objekta. Podaci o neravninama objekta spremljeni su u posebnim teksturama u kojima je bojom označena neravnina objekta. Točnije, spremaju se XYZ komponente vektora normale za svaku pojedinu točku modela koje su potrebne za računanje osvjetljenja objekta.

Za ispravan rad preslikavanja neravnina potrebno je implementirati funkciju koja procesira svaku točku objekta. *Ispis 2.1* prikazuje definiciju funkcije procesora točaka (*eng. pixel shader*). Na početku, potrebno je dohvatiti pripadajući vektor normale iz teksture pomoću funkcije `tex2D` nakon čega je učitanu normalu potrebno podesiti na interval [-1, 1] jer su RGB komponente unutar jezika HLSL u rasponu od [0, 1]. Nakon toga, potrebno je izračunati takozvani polu-vektor h koji za svoj izračun koristi vektor usmjeren prema izvoru svjetlosti (vektor l) i vektor usmjeren prema promatraču (vektor v). Nakon toga, točku sjenčamo Phongovim

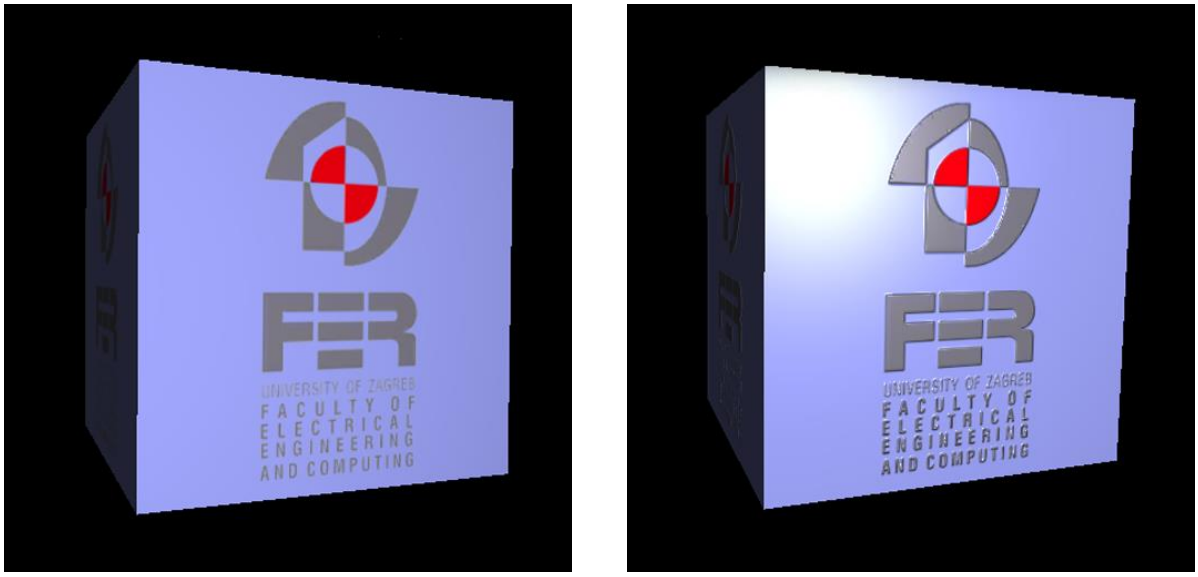
modelom osvjetljenja koji za određivanje boje piksela treba imati informaciju o ambijentalnoj, difuznoj i reflektirajućoj komponenti.

```
float4 Pix(VertexToPixel IN) : COLOR
{
   float3 textureNormal = tex2D(normalMap, IN.texCoord);
   float3 n = normalize(2 * textureNormal - float3(1, 1, 1));

   float3 l = normalize(IN.lightDir);
   float3 v = normalize(IN.viewDir);
   float3 h = normalize(IN.lightDir + IN.viewDir);

   float nDotL = max(dot(n, l), 0);

   float nSpecL = pow(dot(n, h), material.shininess);

   float4 color = (material.ambient * light.ambient) +
      (material.specular * light.specular * nSpecL) +
      (material.diffuse * light.diffuse * nDotL );

   return color * tex2D(colorMap, IN.texCoord);
}
```

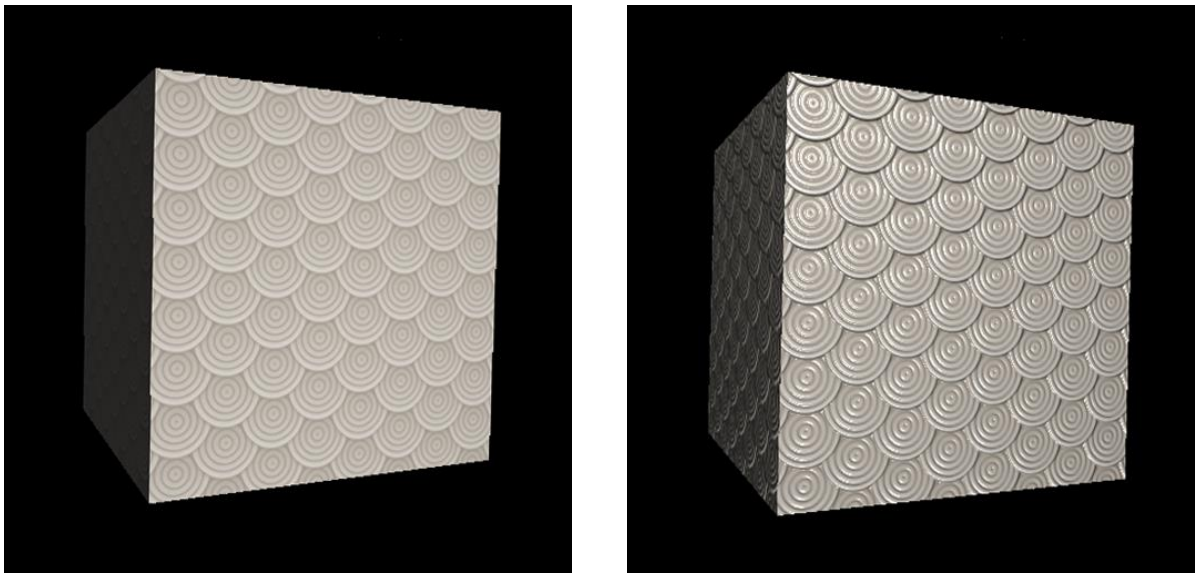
Ispis 2.1: Definicija funkcije procesora točaka

Slika 2.1 prikazuje objekte (kocke) s implementacijom preslikavanja neravnina i bez nje. Kao što se može vidjeti, objekti koji imaju uključeno preslikavanje ravnine izgledaju mnogo bolje od onih objekata koji to nemaju. Važno je napomenuti da svi objekti prikazani na slici imaju jednak broj poligona.

Bez preslikavanja ravnina S preslikavanjem ravnina



Bez preslikavanja ravnina S preslikavanjem ravnina



Slika 2.1: Preslikavanje neravnina na modelu kocke

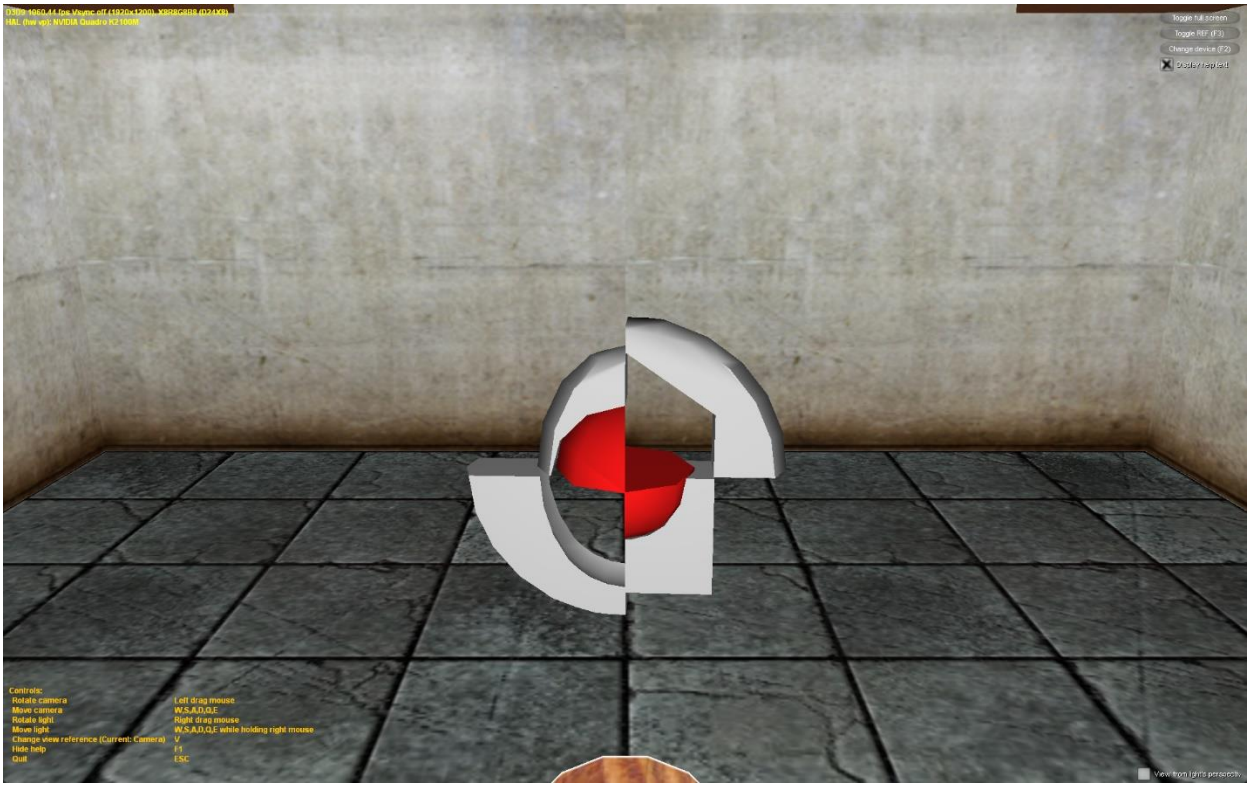
3. Preslikavanje sjena

Preslikavanje sjena (*eng. shadow mapping*) je tehnika stvaranja sjena u kojoj se scena iscrtava iz perspektive svjetla (ili više njih ako postoje) na način da se stvara tekstura u kojoj vrijednost pojedinog slikovnog elementa predstavlja najveću dubinu u sceni iz perspektive tog svjetla. Nakon svih prolaza, dobivene informacije se koriste pri generiranju konačnog prikaza na način da se za svaki slikovni element provjerava je li u sjeni ili nije.

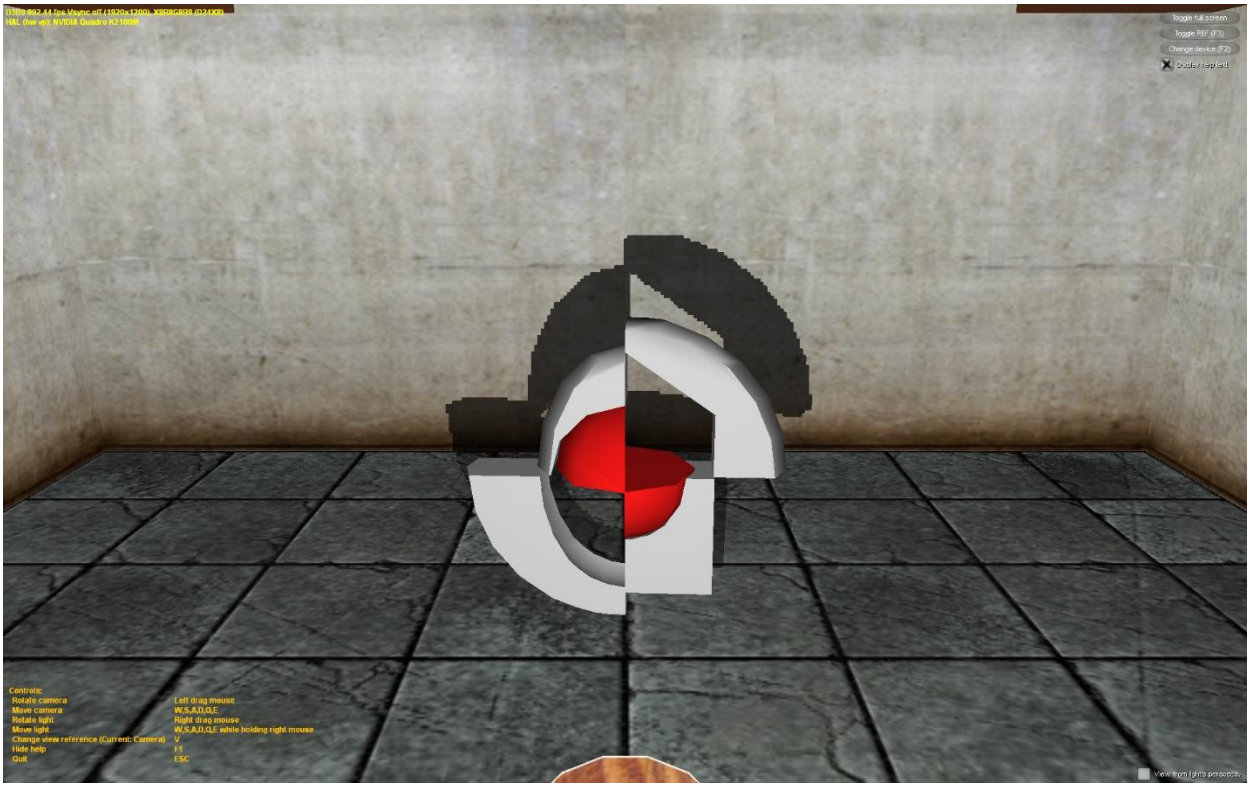
```
float4 PixScene(SceneVertexToPixel IN) : COLOR
{
   float4 color;
   float3 posLightDir = normalize(float3(IN.viewPos - light.pos));
   if(dot(posLightDir, light.dir) > light.cosTheta)
   {
      float3 n = normalize(IN.normal);
      float nDotL = dot(-posLightDir,n);

      float4 lightTransformed = ((IN.viewPosLight /
IN.viewPosLight.w) + float4(1, 1, 1, 1)) / 2;

      lightTransformed.y = 1.0 - lightTransformed.y;

      float shadowDepth = tex2D(shadowMap, lightTransformed.xy).x;

      if (shadowDepth < (IN.viewPosLight.z / xMaxDepth - 0.01)){
         color = light.ambient * material;
      } else
      {
         color = (light.ambient + light.diffuse * nDotL) * material;
      }

   } else
   {
      color = light.ambient * material;
   }
   return  color * tex2D( colorMap, IN.texCoord);
}
```

Ispis 3.1: Definicija funkcije procesora točaka scene korištenjem mape sjena

Prikaz scene se sastoji od dva prolaza. Prvi prolaz generira mapu sjena za postavljeni izvor svjetlosti, dok drugi prolaz generira konačni prikaz cijele scene. *Ispis 3.1* prikazuje definiciju funkcije procesora točaka za scenu korištenjem mape sjena. Prvo što funkcija provjerava jest je li se slikovni element nalazi u osvijetljenom dijelu scene. Ako se nalazi, funkcija tada računa poziciju (njegovu dubinu u sceni) iz perspektive koordinatnog sustava izvora svjetlosti. Ako je izračunata dubina veća od one zapisane u mapi sjena, slikovni element se nalazi u sjeni i potrebno ga je sjenčati samo ambijentalnom komponentom. Navedena provjera ima maleni posmak u iznosu od 0.01 kako ne bi došlo do pojave samo-sjenčanja (*eng. self-shadowing*). S druge strane, ako je dubina slikovnog elementa veća od one zapisane u mapi sjena tada ga treba sjenčati Phongovim modelom osvjetljenja.



Slika 3.1: Scena bez preslikavanja sjena



Slika 3.2: Scena s preslikavanjem sjena

Slika 3.1 prikazuje scenu bez korištenja preslikavanja sjena, dok *Slika 3.2* prikazuje istu scenu s korištenjem preslikavanja sjena. Može se uočiti kako je dobivena sjena vrlo niske kvalitete na rubovima što je rezultat nedovoljno velike rezolucije mape sjena. Ovaj problem je moguće riješiti povećanjem rezolucije mape sijena i dodatnom nadopunom algoritma sjenčanja koji bi sjenčao ovisno o udaljenosti od ruba.