


Računarska grafika

JavaFX – materijal



Pojam materijala

- Na prikaz (izgled) površi objekta utiču:
 - njegova geometrija
 - položaj i orijentacija kamere
 - svetla kojima je obasjan
 - **osobine materijala površi objekta**
- JavaFX omogućava definisanje osobina materijala 3D objekata
 - apstraktna klasa `Material` apstrahuje osobine materijala
- Ako je definisan objekat `Shape3D obj` i materijal `Material mat` materijal objekta može da se postavi pozivom:

```
obj.setMaterial(mat);
```
- Materijali su deljivi između više čvorova tipa `Shape3D`

Fongov materijal

- U biblioteci JavaFX jedini konkretan materijal je `PhongMaterial`
 - klasa je izvedena iz klase `Material`
- Obe klase pripadaju paketu `javafx.scene.paint`
- Na materijal tipa `PhongMaterial` se primenjuje:
 - Fongov model osvetljenja (koji podržava i efekat refleksije materijala) i
 - Fongovo senčenje
- Fongov materijal ima i druga interesantna svojstva
 - lista je data na sledećem slajdu

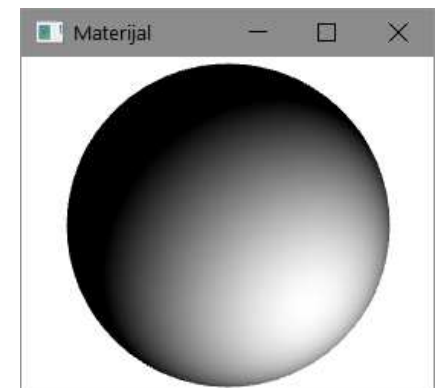
Svojstva klase PhongMaterial

- Klasa `PhongMaterial` ima sledeća svojstva:
 - difuzna boja: `diffuseColor`
 - mapa difuzije (tekstura): `diffuseMap`
 - boja odsjaja: `specularColor`
 - snaga odsjaja: `specularPower`
 - mapa odsjaja: `specularMap`
 - mapa reljefa: `bumpMap`
 - mapa samo-osvetljenja: `selfIlluminationMap`

Primer

```
Sphere lopta = new Sphere(100);  
// Pozicioniranje lopte  
PhongMaterial mat = new PhongMaterial();  
// Ovde će da bude dodavan kod  
// koji upravlja atributima materijala  
lopta.setMaterial(mat);
```

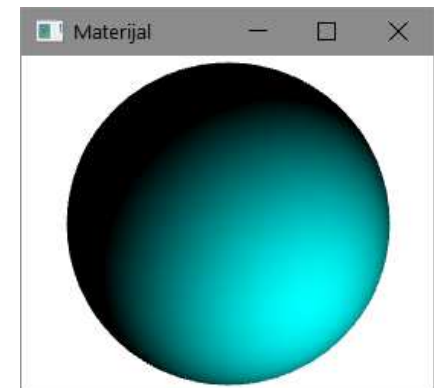
```
PointLight svetlo = new PointLight();  
// Pozicioniranje svetla  
Group koren = new Group(lopta, svetlo);
```



Difuzna boja

- Difuzna boja (`DiffuseColor`)
 - boja materijala od kojeg se svetlo difuzno odbija
 - utiču i izvori ambijentalnog svetla i tačkasti izvori svetla
 - na svetlo od tačkastog izvora se primenjuje model difuznog osvetljenja
- Difuzna boja određuje boju difuzne komponente odbijene svetlosti kada je objekat obasjan samo belom svetlošću
- Ako se postavi samo ovo svojstvo materijala
 - objekat je monolitno obojen, uz odgovarajuće senčenje
- Primer:

```
mat.setDiffuseColor(Color.CYAN);
```



Mapa difuzije

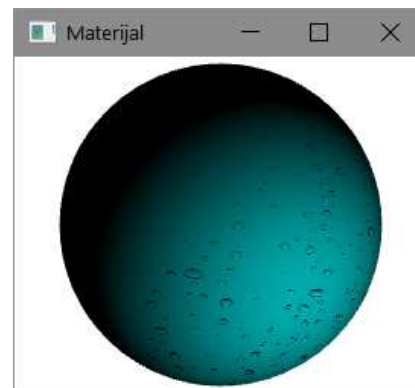
- Mapa difuzije (`diffuseMap`)
 - pravougaona tekstura
 - može da se formira na osnovu učitane slike
 - slika se preslikava na trouglove mreže objekta



```
mat.setDiffuseMap(  
new Image("kapi.jpg"));
```



```
mat.setDiffuseColor(Color.CYAN);  
mat.setDiffuseMap(new Image("kapi.jpg"));
```



Odsjaj

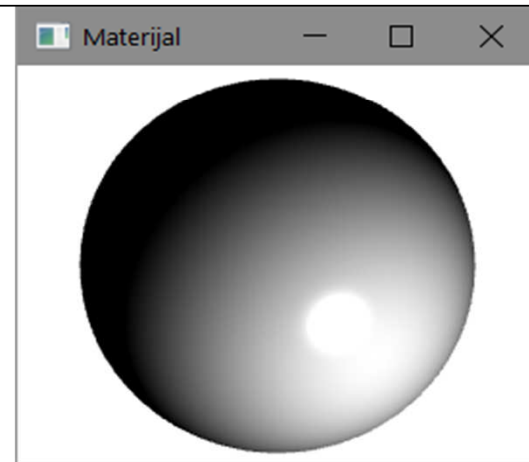
- Komponenta odbijene svetlosti koja predstavlja odsjaj svetosnog izvora od površi objekta
- Reflektivna (spekularna) komponenta osvetljenja
- Računa se na osnovu Fongovog modela osvetljenja
- Komponenta odsjaja je opisana svojstvima:
 - boja odsjaja (`specularColor`)
 - snaga odsjaja (`specularPower`)
 - mapa odsjaja (`specularMap`)

Mapa odsjaja

- Mapa odsjaja (`specularMap`) omogućava da različiti delovi površi objekta imaju različitu refleksionu moć
- Kao i za mapu difuzije, za mapu odsjaja se koristi tekstura formirana na osnovu učitane slike
- Boja piksela na mapi odsjaja utiče na boju piksela površi objekta
 - crna boja potpuno sprečava odsjaj
 - bela omogućava puni intenzitet odsjaja
- Mapa odsjaja se preslikava na trouglove koji čine mrežu objekta na isti način kao mapa difuzije

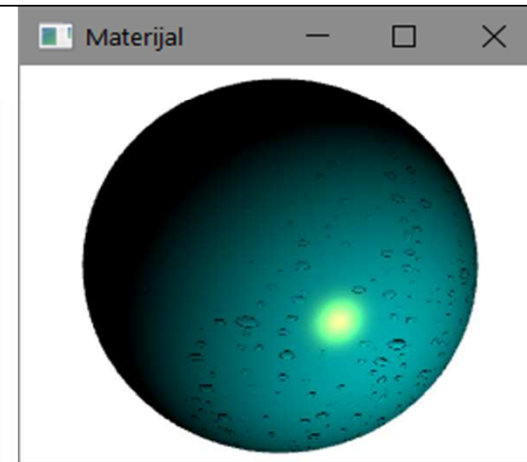
Boja i mapa odsjaja - primer

```
mat.setSpecularColor(  
Color.WHITE);
```

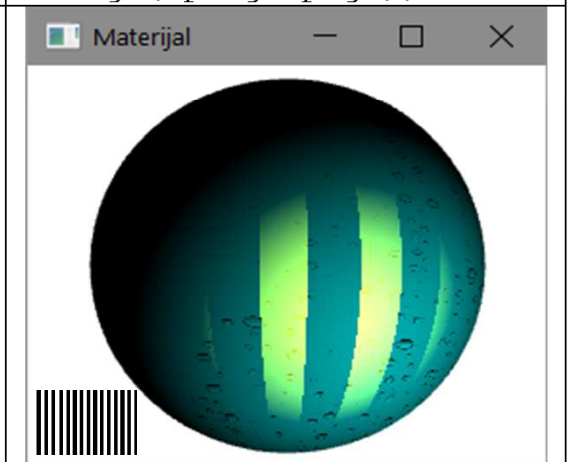


SP=32

```
mat.setDiffuseColor(  
Color.CYAN);  
mat.setDiffuseMap(new  
Image("kapi.jpg"));  
mat.setSpecularColor(  
Color.YELLOW);
```



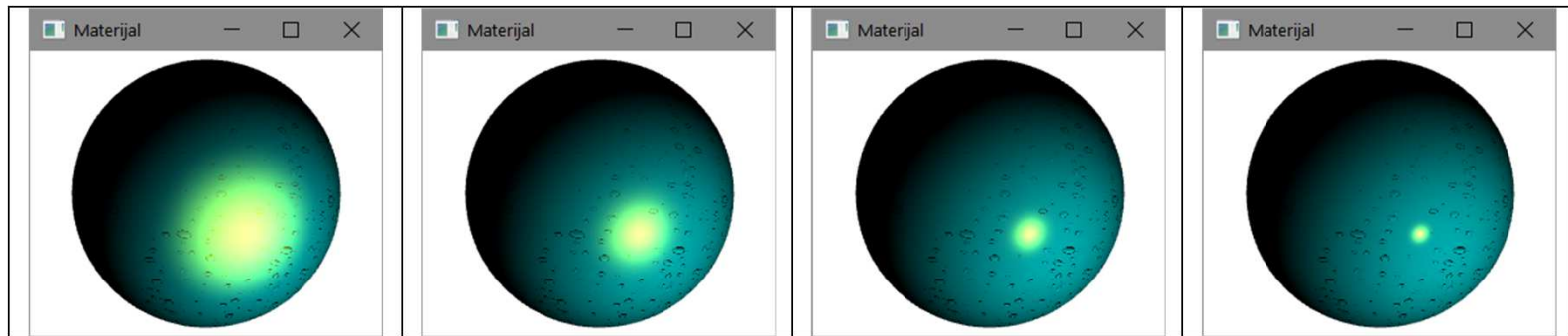
```
mat.setDiffuseColor(  
Color.CYAN);  
mat.setDiffuseMap(new  
Image("kapi.jpg"));  
mat.setSpecularColor(  
Color.YELLOW);  
mat.setSpecularPower(1);  
mat.setSpecularMap(new  
Image("pruge.png"));
```



SP=1

Snaga odsjaja

- Što veća snaga, to bolji fokus odsjaja
- U formuli za komponentu odsjaja $(R \cdot V)^n$:
$$n = (\text{specularPower} * \text{intenzitet}(\text{specularMap}))$$
- Podrazumevana snaga odsjaja je SP=32



SP=2

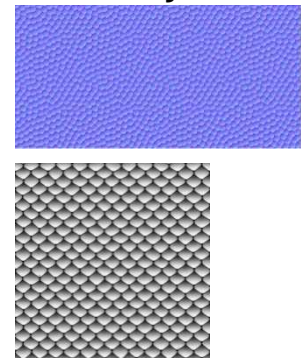
SP=8

SP=32


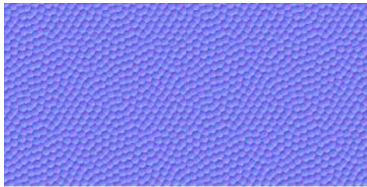
SP=128

Mapa reljefa (neravnina)

- Mapa reljefa (`bumpMap`) se koristi za vizuelni efekat neravne površi objekta
- Sadrži matricu normala, da bi se efektom osvetljenja proizveo vizuelni efekat udubljenja/ispupčenja na površi objekta
- Kao i prethodne mape i ova se može da se formira od učitane slike
 - RGB vrednosti piksela → komponente jediničnih vektora normale u datoj tački posle preslikavanja mape na trougaonu mrežu objekta
 - slika je u plavičastim nijansama
- Druga varijanta mape reljefa (JavaFX ne podržava):
 - vertikalno odstupanje tačke od horizontalne površi
 - slika je u skali sive: bela – maks. ispupčenje, crna – maks. udubljenje



Mapa reljefa zadanog normalama

- Svaka komponenta jediničnog vektora \rightarrow vrednost u rasponu $[-1, 1]$
- Odgovarajuće komponente boje (r,g,b), u rasponu $[0, 1]$:
 $r=(x+1)/2$, $g=(y+1)/2$, $b=(z+1)/2$
- Vektor koji je normalan na XoY površ ima vrednost $(0, 0, 1)$
 - vrednosti RGB boje za njega: $(0.5, 0.5, 1)$ 
 - piksel odgovarajuće slike ima najintenzivniju plavu komponentu
- Vektori svih normala, osim u tačkama gde postoje neravnine u odnosu na geometrijsku površ, treba da budu $(0, 0, 1)$,
 - razlog zašto su slike koje predstavljaju mape neravnina zadate normalama u plavičastim nijansama 

Primer – mapa reljefa Zemlje

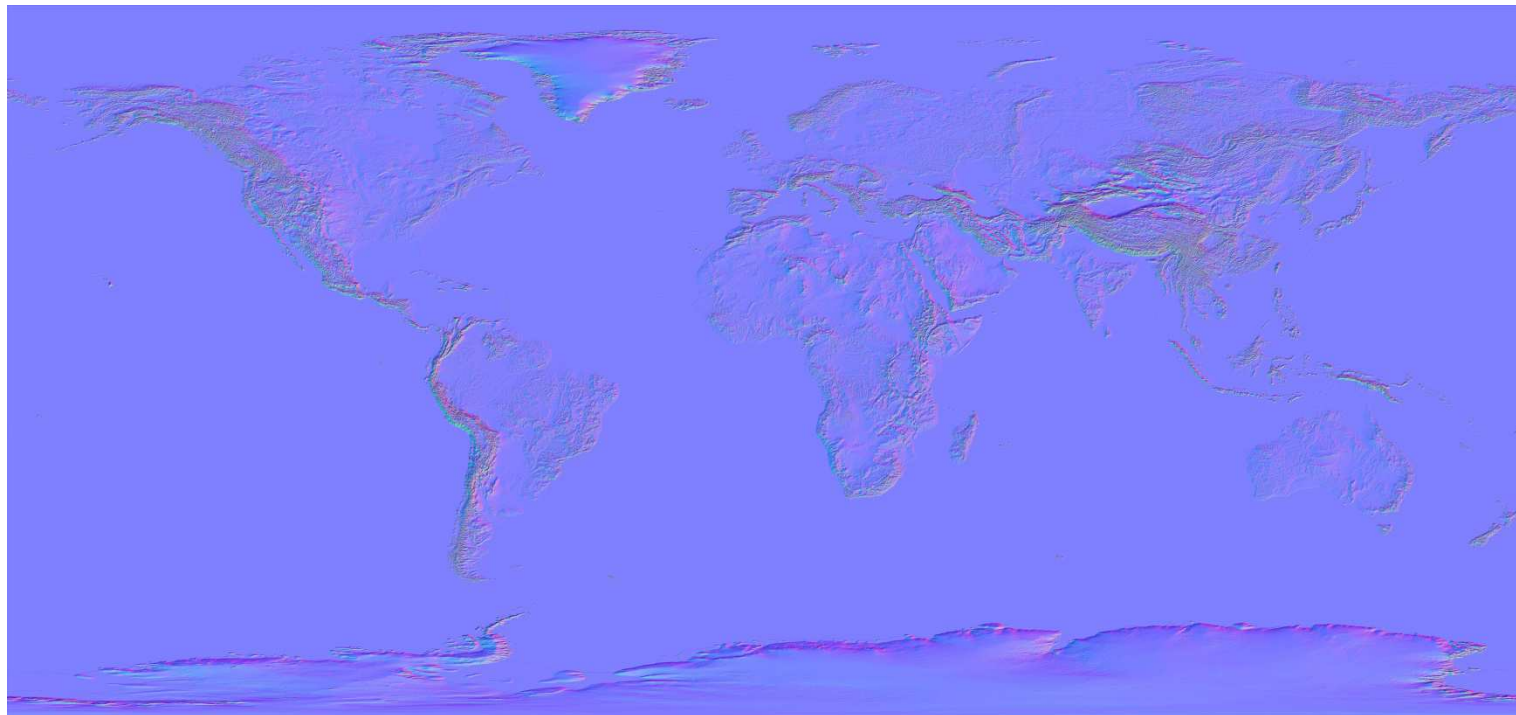
(0.5,0.5,1)



(0.5,1,0.5)

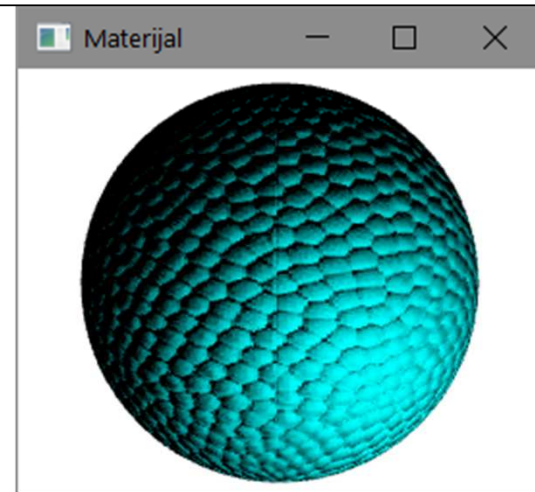


(1,0.5,0.5)

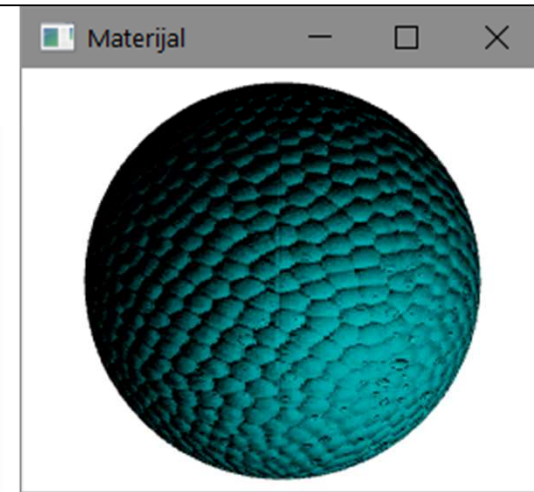


Mapa reljefa – primer

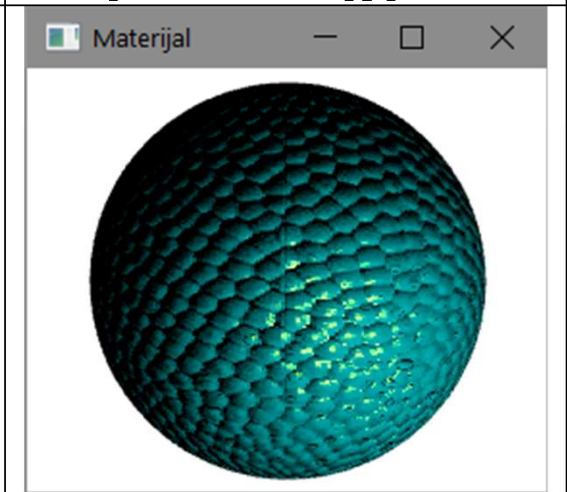
```
mat.setDiffuseColor(  
    Color.CYAN);  
mat.setBumpMap(new  
    Image("normale.jpg"));
```



```
mat.setDiffuseColor(  
    Color.CYAN);  
mat.setDiffuseMap(new  
    Image("kapi.jpg"));  
mat.setBumpMap(new  
    Image("normale.jpg"));
```

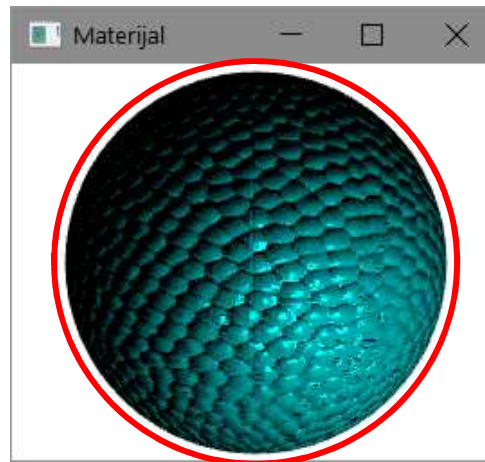


```
mat.setDiffuseColor(  
    Color.CYAN);  
mat.setDiffuseMap(new  
    Image("kapi.jpg"));  
mat.setSpecularColor(  
    Color.YELLOW);  
mat.setBumpMap(new  
    Image("normale.jpg"));
```



Nedostatak mape neravnina

- Primenom mape neravnina ne menja se geometrija objekta
- Samo se postiže vizuelni efekat osvetljaja na njegovoj površi
- Da se ne menja geometrija može da se uoči posmatranjem konture
 - na njoj ne postoje neravnine u odnosu na model



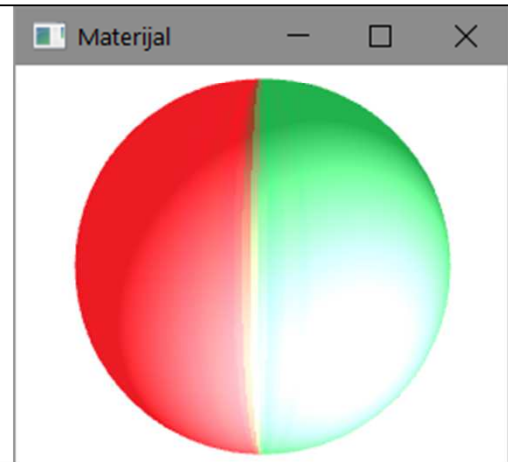
JavaFX - materijal

Samo-osvetljenje

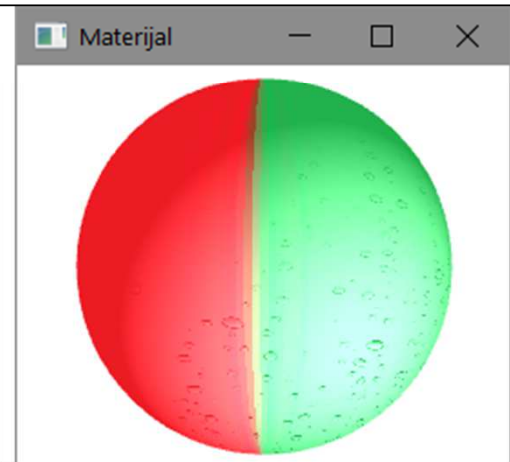
- Mapa samo-osvetljenja (`selfIlluminationMap`)
 - koristi se za objekte koji su iznutra osvetljeni, kao što su lampe
- Mapa određuje treću komponentu osvetljenosti tačke površi objekta (osim difuzne i komponente odsjaja)
- Difuzna i komp. odsjaja određuju boju na osnovu odbijene svetlosti
- Komp. samo-osvetljenja potiče od zračenja samog objekta
 - odnosno simulirane propuštene svetlosti iz unutrašnjosti objekta
- Koristi se slika čiji pikseli, posle preslikavanja na mrežu objekta, određuju boju i intenzitet samo-osvetljaja u datoj tački površi
- Tamnije nijanse isijavaju manje sopstvenog svetla

Samo-osvetljenje – primer

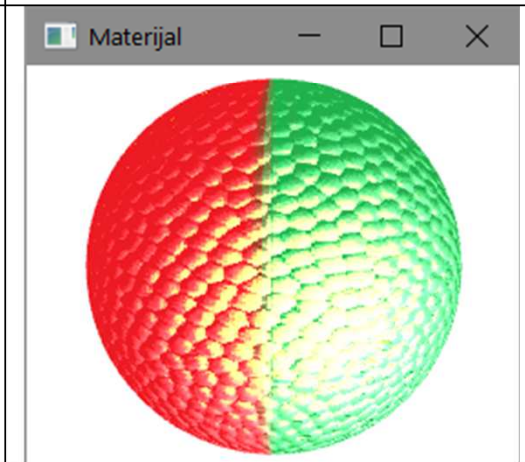
```
mat.setSelfIlluminationMap  
(new Image("cr_zel.jpg"));
```



```
mat.setDiffuseMap(new  
Image("kapi.jpg"));  
mat.setSelfIlluminationMap  
(new Image("cr_zel.jpg"));
```



```
mat.setDiffuseMap(new  
Image("kapi.jpg"));  
mat.setSpecularPower(1);  
mat.setSpecularColor(  
Color.YELLOW);  
mat.setBumpMap(new  
Image("normale.jpg"));  
mat.setSelfIlluminationMap  
(new Image("cr_zel.jpg"));
```



Osvetljenje – boja u tački

- Sledeći pseudo-kod predstavlja algoritam izračunavanja boje u tački

```
za svaki izvor i ambijentalnog svetla {
    ambient += lightColor[i]
}
za svaki izvor i tačkastog svetla {
    diffuse += (L[i] · N) * lightColor[i]
    specular += ((R[i] · V) ^
                (specularPower * intensity(specularMap)))
                * lightColor[i]
}
color = (ambient + diffuse) * diffuseColor * diffuseMap
        + specular * specularColor * specularMap
        + selfIlluminationMap
```

Napomena

- Voditi računa da materijal nije modalni atribut
 - promena svojstava materijala se primenjuje i na objekte kojima je prethodno postavljen taj materijal

- Primer:

```
PhongMaterial mat = new PhongMaterial();  
mat.setDiffuseMap( new Image("tekstura1.jpg"));  
podloga.setMaterial(mat);  
mat.setDiffuseMap( new Image("tekstura2.png"));  
objekat.setMaterial(mat);
```

- Posledica:

- i podloga i objekat će da dobiju istu teksturu `tekstura2.png`