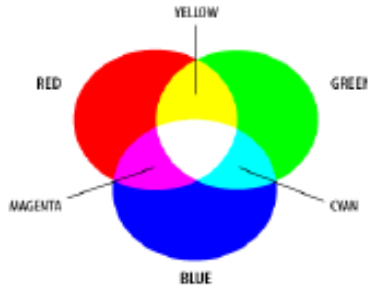
Osvetljenje

Računarska grafika

Boje



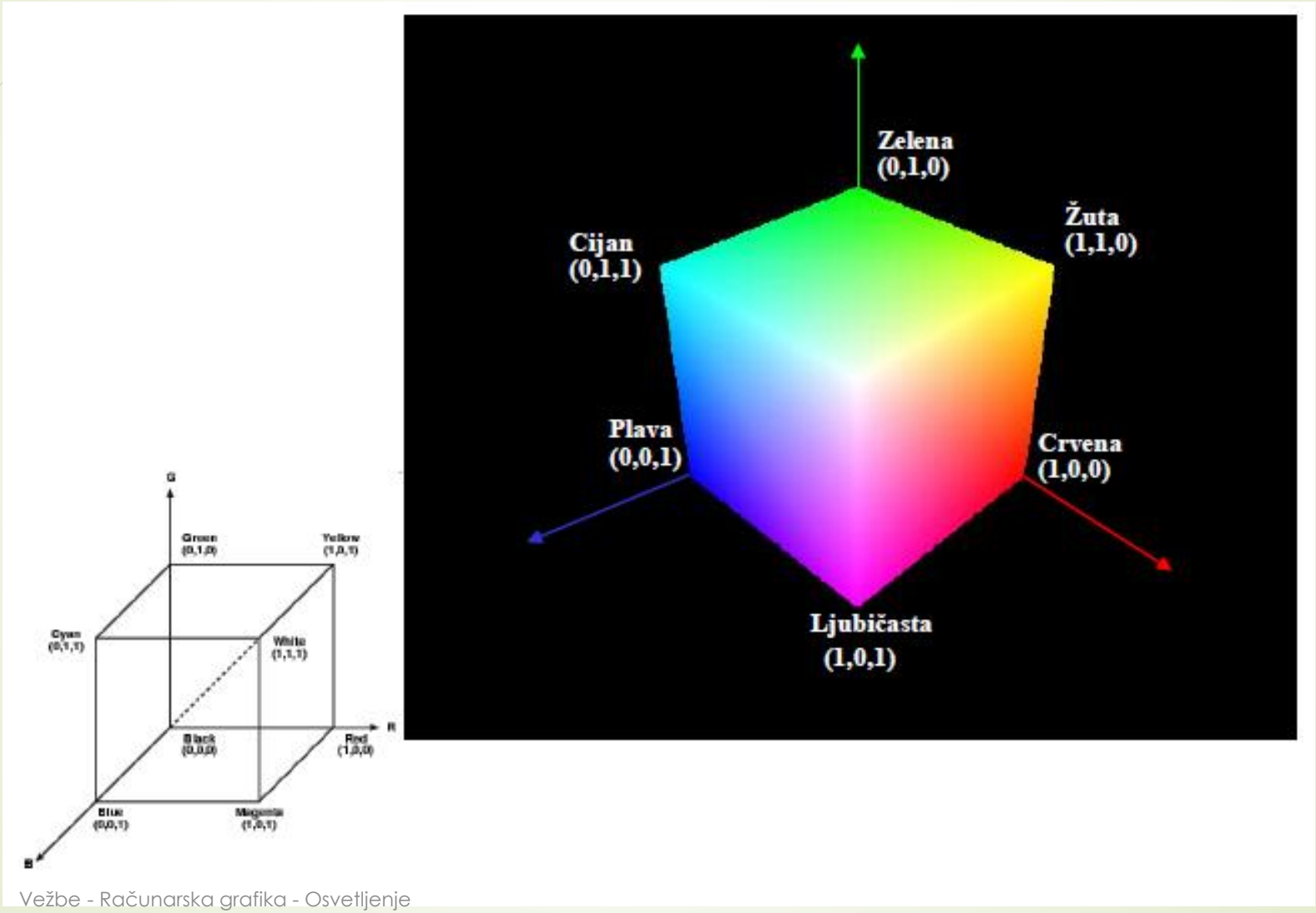


Boja je senzacija koju svetlosna energija izaziva na retini i koji se interpretira u mozgu.

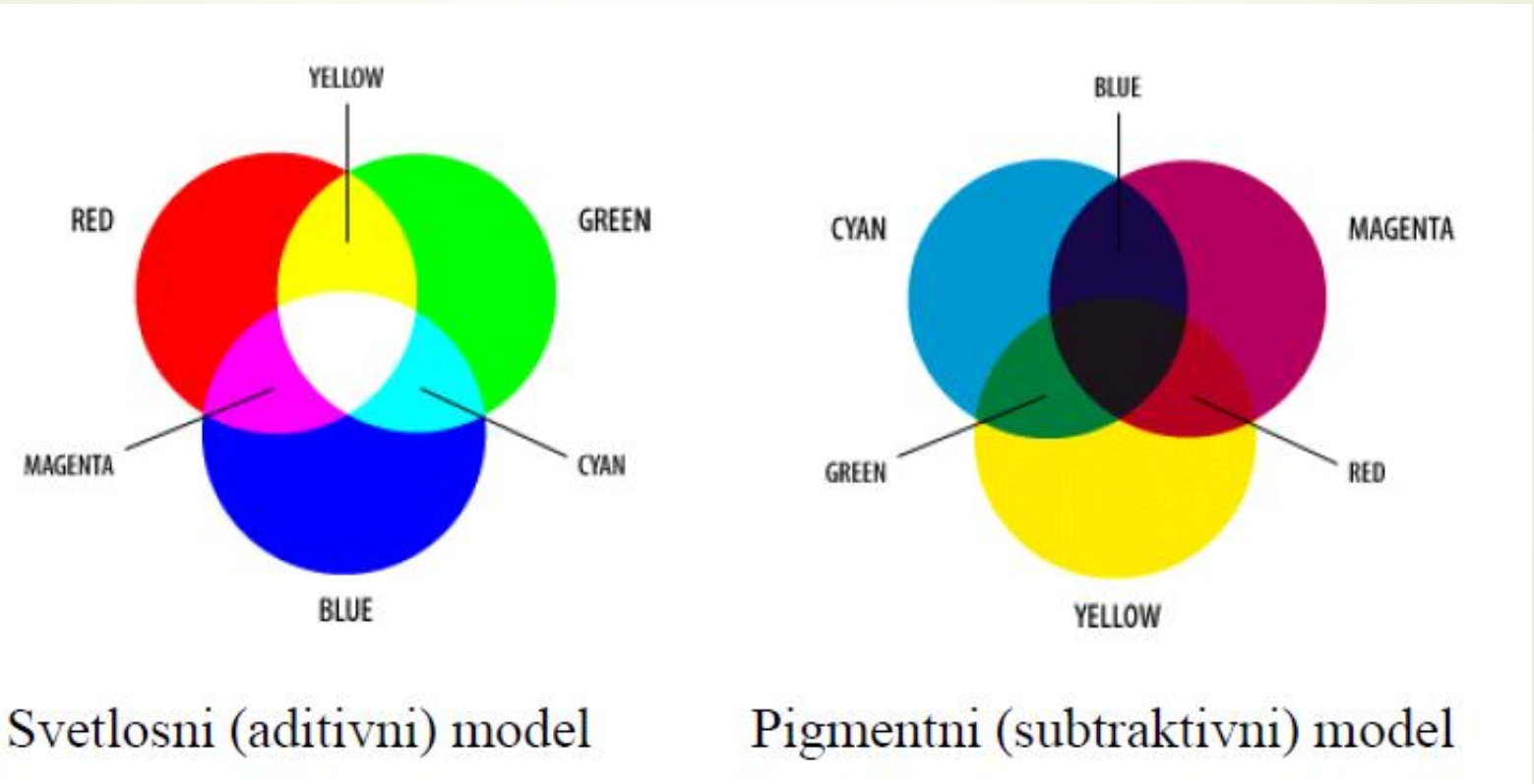
Mozak formira sliku na osnovu mešavine fotona različitih frekvencija.

Ljudi poseduju 3 tipa konusnih ćelija, koje reaguju na kratke, srednje i duge talasne dužine (crvena, zelena i plava).

RGB Model


Zelena
(0,1,0)
Žuta
(1,1,0)
Cijan
(0,1,1)
Plava
(0,0,1)
Crvena
(1,0,0)
Ljubičasta
(1,0,1)
G
R
B
Green
(0,1,0)
Yellow
(1,0,1)
Cyan
(0,1,1)
White
(1,1,1)
Black
(0,0,0)
Red
(1,0,0)
Blue
(0,0,1)
Magenta
(1,0,1)

Mešanje boja



Svetlosni (aditivni) model

Pigmentni (subtraktivni) model

Zadavanje boje

void glColor3{b s i f d ub us ui} (TYPE *r*, TYPE *g*, TYPE *b*);

void glColor4{b s i f d ub us ui} (TYPE *r*, TYPE *g*, TYPE *b*, TYPE *a*);

void glColor3{b s i f d ub us ui}v (const TYPE **v*);

void glColor4{b s i f d ub us ui}v (const TYPE **v*);

Suffix	Data Type	Minimum Value	Min Value Maps to	Maximum Value	Max Value Maps to
b	1-byte integer	-128	-1.0	127	1.0
s	2-byte integer	-32,768	-1.0	32,767	1.0
i	4-byte integer	-2,147,483,648	-1.0	2,147,483,647	1.0
ub	unsigned 1-byte integer	0	0.0	255	1.0
us	unsigned 2-byte integer	0	0.0	65,535	1.0
ui	unsigned 4-byte integer	0	0.0	4,294,967,295	1.0

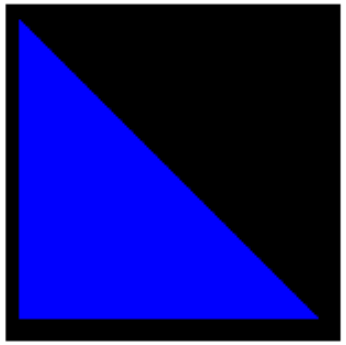
Izbor načina senčenja

void **glShadeModel** (GLenum *mode*);

- GL_SMOOTH (default)
- GL_FLAT



glShadeModel(GL_SMOOTH);



glShadeModel(GL_FLAT);

```
glBegin (GL_TRIANGLES);
    glColor3f (1.0, 0.0, 0.0);
    glVertex3f (0.0, 0.0, 0.0);
    glColor3f (0.0, 1.0, 0.0);
    glVertex3f (2.0, 0.0, 0.0);
    glColor3f (0.0, 0.0, 1.0);
    glVertex3f (0.0, 2.0, 0.0);
glEnd ();
```

Koraci za dodavanje osvetljenja

- definisanje normala u svakom od temena modela
- kreiranje, selekcija i pozicioniranje jednog ili više izvora svetlosti
- izbor modela osvetljenja
- definisanje svojstava materijala objekata u sceni

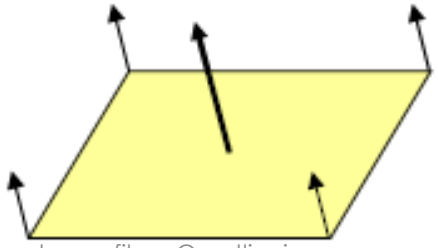
Normale

Normala se može definisati samo u temenima i zadaje se neposredno pre *glVertex* komande.

void **glNormal3**{bsidf}(*TYPEnx*, *TYPEny*, *TYPEnz*);
void **glNormal3**{bsidf}**v**(const **TYPE** *v);

Ako se koriste transformacije koje narušavaju normalizaciju normale (Scale, Shear, ...), treba omogućiti automatsku normalizaciju pozivom:

glEnable(GL_NORMALIZE)



```
glBegin (GL_POLYGON);
   glNormal3fv(n0);
   glVertex3fv(v0);
   glNormal3fv(n1);
   glVertex3fv(v1);
   glNormal3fv(n2);
   glVertex3fv(v2);
   glNormal3fv(n3);
   glVertex3fv(v3);
glEnd();
```

Normale parametarskih površi

Ako je površ zadata analitički: $V(s,t) = [X(s,t)\ Y(s,t)\ Z(s,t)]$

gde su X, Y i Z diferencijabilne funkcije po **s** i **t**.

Treba naći $\frac{\partial V}{\partial s}$ i $\frac{\partial V}{\partial t}$ a zatim $\frac{\partial V}{\partial s} \times \frac{\partial V}{\partial t}$

Pacijalni izvodi po **s** i **t** daju vektore tangente na površ, po **s** i **t** pravcu, vektorski proizvod je upravan na oba, pa time i na površinu.

Vektorski proizvod dva vektora $V(v_x, v_y, v_z)$ i $W(w_x, w_y, w_z)$, računa se po formuli:

$$V \times W = [v_x\ v_y\ v_z] \times [w_x\ w_y\ w_z] = [(v_y w_z - v_z w_y)\ (v_z w_x - v_x w_z)\ (v_x w_y - v_y w_x)].$$

Normalizacija (svođenje na jedinični vektor): deljenje svake komponente sa dužinom

$$len = \sqrt{x^2 + y^2 + z^2}$$

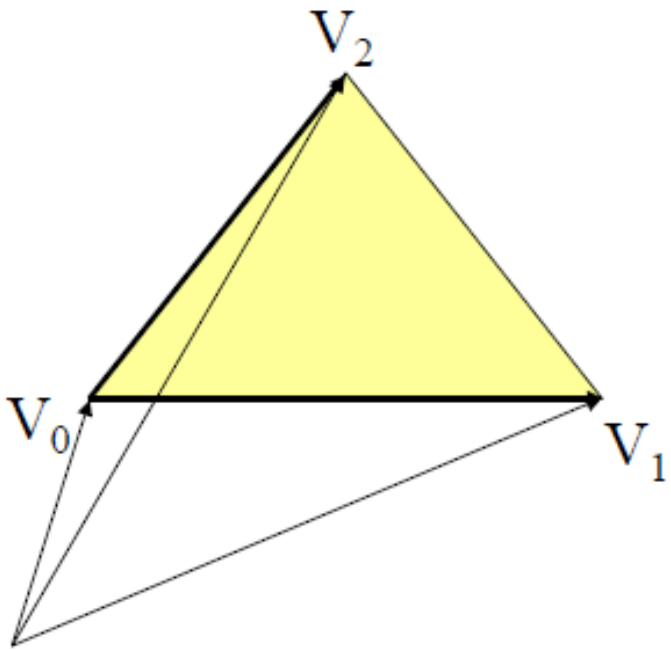
Primer: Ako je V(s,t) = [s^2 t^3 3-st]

$$\frac{\partial V}{\partial s} = [2s\ 0\ -t],\ \frac{\partial V}{\partial t} = [0\ 3t^2\ -s],\ \text{and}\ \frac{\partial V}{\partial s} \times \frac{\partial V}{\partial t} = [-3t^3\ 2s^2\ 6st^2]$$

Posle normalizacije: (-24/34, 2/34, 24/34) = (-0.70588, 0.058823, 0.70588)

Normale poligonalnih površi

$$[v_x\ v_y\ v_z] \times [w_x\ w_y\ w_z] = [(v_y w_z - v_z w_y)\ (v_z w_x - v_x w_z)\ (v_x w_y - v_y w_x)]$$

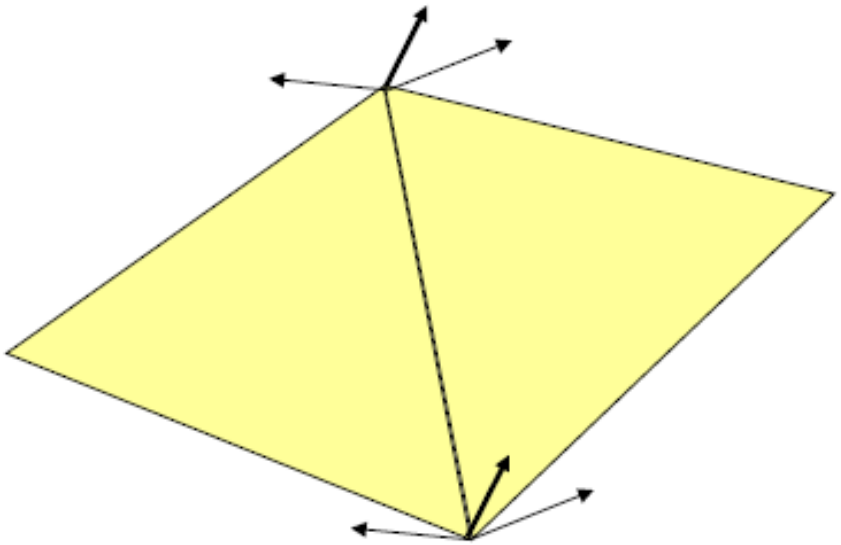


$$N = [V_1 - V_0] \times [V_2 - V_0]$$

```
void normcrossprod(float v1[3], float v2[3], float out[3])
{
    GLint i, j;
    GLfloat length;
   out[0] = v1[1]*v2[2] - v1[2]*v2[1];
   out[1] = v1[2]*v2[0] - v1[0]*v2[2];
   out[2] = v1[0]*v2[1] - v1[1]*v2[0];
   normalize(out);
}

void normalize(float v[3]) {
GLfloat d = sqrt(v[1]*v[1]+v[2]*v[2]+v[3]*v[3]);
    if (d == 0.0) {
      error("zero length vector");
          return;
   }
   v[1] /= d; v[2] /= d; v[3] /= d;
}
```

Simulacija zakrivljenih površi



Za zakrivljene površine koristiti istu vrednost (u datom temenu) za normale svih poligona koji dele dato teme. Orijentacija normale dobija se kao srednja vrednost normala susednih površinu.

Komponente osvetljenja



- emisiona – potiče od objekta i na nju ne utiču ostali izvori
- ambijentalna – više puta reflektovana svetlost, tako da se ne može oceniti pravac
- difuziona – dolazi iz jednog pravca ali se rasipa na sve strane, tako da objekat izgleda jednako osvetljen bez obzira gde je posmatrač (zavisi samo od ugla svetla i normale na površinu – Lambertov "kosinusni" zakon)
- refleksiona – dolazi iz određenog pravca i reflekuje se od sjajnih površina

Ako dva izvora svetlosti emituju **(R1,G1,B1)** i **(R2,G2,B2)**, rezultujuće svetlo biće **(R1+R2,G1+G2,B1+B2)**. Pokomponentni zbir se odseca na vrednost 1.0.

Ako je boja materijala **(MR,MG,MB)** i obasjan je svetlošću **(LR,LG,LB)**, rezultujuća boja biće **(MR*LR,MG*LG,MB*LB)**.

Kreiranje izvora svetlosti

Svetla imaju mnoštvo parametara: boju, poziciju, usmerenje, ... i svi oni postavljaju se komandom **glLight*()**.

void glLight{if}[v](GLenum *light*, GLenum *pname*, *TYPE param*);

***light* može biti: GL_LIGHT0, GL_LIGHT1, ... , ili GL_LIGHT7**

Primer:

```
GLfloat light_ambient[] = { 0.0, 0.0, 0.0, 1.0 };
GLfloat light_diffuse[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat light_specular[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat light_position[] = { 1.0, 1.0, 1.0, 0.0 };
glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);
glLightfv(GL_LIGHT0, GL_POSITION, light_position);
```

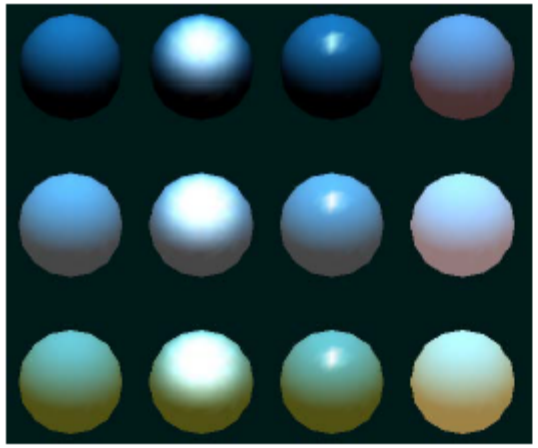
Parametri svetla i njihove vrednosti

Parametar	Podrazumevana vrednost	Značenje
GL_AMBIENT	(0.0, 0.0, 0.0, 1.0)	Intenzitet ambijentalne RGBA kom
GL_DIFFUSE	(1.0, 1.0, 1.0, 1.0)	Intenzitet difuzione RGBA kom.
GL_SPECULAR	(1.0, 1.0, 1.0, 1.0)	Intenzitet refleksione RGBA kom.
GL_POSITION	(0.0, 0.0, 1.0, 0.0)	(x, y, z, w) pozicija svetla
GL_SPOT_DIRECTION	(0.0, 0.0, -1.0)	(x, y, z) usmerenje izvora
GL_SPOT_EXPONENT	0.0	Eksponent usmerenog izvora
GL_SPOT_CUTOFF	180.0	Ugao kupe usmerenog izvora
GL_CONSTANT_ATTENUATION	1.0	Faktor konstantne atenuacije
GL_LINEAR_ATTENUATION	0.0	Faktor linearne atenuacije
GL_QUADRATIC_ATTENUATION	0.0	Faktor kvadratne atenuacije

Boja

OpenGL omogućuje definisanje 3 parametra vezana za boju, i to:

- ambijentalnu komponentu (GL_AMBIENT) (obično == GL_DUFFUSE)
- difuzionu komp. (GL_DUFFUSE) – to je ono što zapažamo kao boju svetla i
- refleksionu komp. (GL_SPECULAR) (obično == (1.0,1.0,1.0) ili GL_DUFFUSE)



Pozicija i atenuacija

Izvori mogu biti:

- direkcioni GLfloat light_position[] = { 1.0, 1.0, 1.0, 0.0 };

  Svi zraci koji dopiru do objekata smatraju se paralelnim (npr. sunce je takav izvor)

- pozicioni GLfloat light_position[] = { 1.0, 1.0, 1.0, 1.0 };

  Ugao zraka zavisi od položaja svetla u sceni (npr. lampa). Zahteva mnogo više izračunavanja, pa ukoliko nije neophodno treba koristiti direkciono svetlo.

Kod realnih izvora, intenzitet svetla opada sa rastojanjem. Ovo ima smisla samo kod pozicionih svetala. Faktor slabljenja za poziciona svetla računa se po formuli:

$$faktor_atenuacije = \frac{1}{k_c + k_l d + k_q d^2}$$

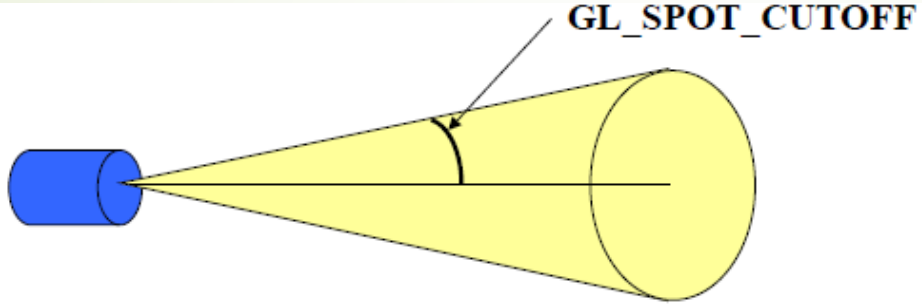
d – rastojanje između izvora svetla i temena objekta

k_c = GL_CONSTANT_ATTENUATION

k_l = GL_LINEAR_ATTENUATION

k_q = GL_QUADRATIC_ATTENUATION

Usmereni izvori svetlosti



Prostor koji osvetljava poziciono svetlo može se omeđiti kupom definisanjem GL_SPOT_CUTOFF parametra koji definiše ugao između pravca svetla i zraka koji putuje ivicom kupe. Ugao kupe je 2 puta veći. Po defaultu, ovo je isključeno, obzirom da je vrednost postavljena na 180 (kupa je sfera!!!). GL_SPOT_CUTOFF parametra mora biti u opsegu [0.0, 90.0], da bi kupa postojala.

```
GLfloat spot_direction[] = { -1.0, -1.0, 0.0 };
glLightfv(GL_LIGHT0, GL_SPOT_DIRECTION, spot_direction);
glLightf(GL_LIGHT0, GL_SPOT_CUTOFF, 45.0);
```

Distribucijau intenziteta svetlosti u okviru kupe definiše (osim atenuacije i) GL_SPOT_EXPONENT. Intenzitet opada ka obodu kupe sa kosinusom rastojanja od ose kupe. Što je GL_SPOT_EXPONENT veći, to je svetlo usmerenije i brže opada ka rubu kupe. Ako je 0 (po default-u), nema slabljenja.

Primer usmerenog izvora

```
GLfloat light1_ambient[] = { 0.2, 0.2, 0.2, 1.0 };
GLfloat light1_diffuse[] =  { 1.0, 1.0, 1.0, 1.0 };
GLfloat light1_specular[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat light1_position[] = { -2.0, 2.0, 1.0, 1.0 };
GLfloat spot_direction[] =  { -1.0, -1.0, 0.0 };

glLightfv(GL_LIGHT1,  GL_AMBIENT,   light1_ambient);
glLightfv(GL_LIGHT1,  GL_DIFFUSE,    light1_diffuse);
glLightfv(GL_LIGHT1,  GL_SPECULAR, light1_specular);
glLightfv(GL_LIGHT1,  GL_POSITION,   light1_position);
glLightf(  GL_LIGHT1,  GL_CONSTANT_ATTENUATION,  1.5);
glLightf(  GL_LIGHT1,  GL_LINEAR_ATTENUATION,       0.5);
glLightf(  GL_LIGHT1,  GL_QUADRATIC_ATTENUATION, 0.2);

glLightf(GL_LIGHT1,   GL_SPOT_CUTOFF,      45.0);
glLightfv(GL_LIGHT1, GL_SPOT_DIRECTION, spot_direction);
glLightf(GL_LIGHT1,   GL_SPOT_EXPONENT,  2.0);

glEnable(GL_LIGHT1);
```

Upravljanje pozicijom i usmerenjem osvetljenja

- Fiksno osvetljenje
- Pokretno osvetljenje
- Osvetljenje vezano za pogled

Fiksno osvetljenje

Svetlo se (u 3D *pipeline*-u) mora postaviti posle svih *model-view* transformacija, tj. u kodu se mora postaviti pre ostalih *model-view* transformacija, da one ne bi uticale na položaj svetla.

```
glViewport(0, 0, w, h);
glMatrixMode (GL_PROJECTION);
glLoadIdentity();
if (w <= h)
    glOrtho (-1.5, 1.5, -1.5*h/w, 1.5*h/w, -10.0, 10.0);
else
    glOrtho (-1.5*w/h, 1.5*w/h, -1.5, 1.5, -10.0, 10.0);
glMatrixMode (GL_MODELVIEW);
glLoadIdentity();
...
GLfloat light_position[] = { 1.0, 1.0, 1.0, 1.0 };
glLightfv(GL_LIGHT0, GL_POSITION, position); //postavljanje poz. pri inicijal.
```

Pokretno osvetljenje

Svetlo možemo rotirati oko nepokretne scene kao u narednom primeru. Trenutna model-view matrica utiče na poziciju svetla.

```
GLfloat light_position[] = { 0.0, 0.0, 1.5, 1.0 };
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
glPushMatrix();
        glTranslatef(0.0, 0.0, -5.0);
         glPushMatrix();
            glRotated((GLdouble) spin, 1.0, 0.0, 0.0);
            glLightfv(GL_LIGHT0, GL_POSITION, light_position);
        glPopMatrix();
        auxSolidTorus();
glPopMatrix();
glFlush();
```

MoveLight.c (exe)

Osvetljenje vezano za pogled

Slično kao kod fiksnog osvetljenja, osim što mora pružati zrake duž negativne orijentacije Z-ose (obzirom da se nakon svih transformacija posmatrač nalazi u koordinatnom početku i gleda u pravcu negativne Z-ose).

```
GLfloat light_position() = { 0.0, 0.0, 1.0, 1.0 };
glViewport(0, 0, w, h);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(40.0, (GLfloat) w/(GLfloat) h, 1.0, 100.0);
glMatrixMode(GL_MODELVIEW);
glLightfv(GL_LIGHT0, GL_POSITION, light_position);
```

Izbor modela osvetljenja

- globalno ambijentalno svetlo

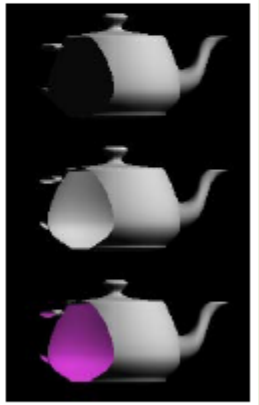
```
GLfloat lmodel_ambient[] = { 0.2, 0.2, 0.2, 1.0 };
glLightModelfv(GL_LIGHT_MODEL_AMBIENT, lmodel_ambient);
```

- lokalni ili udaljeni pogled

```
glLightModeli(GL_LIGHT_MODEL_LOCAL_VIEWER, GL_TRUE); ili GL_FALSE
```

- dvostrano osvetljenje

```
glLightModeli(GL_LIGHT_MODEL_TWO_SIDE, GL_TRUE);
```



- aktiviranje osvetljenja

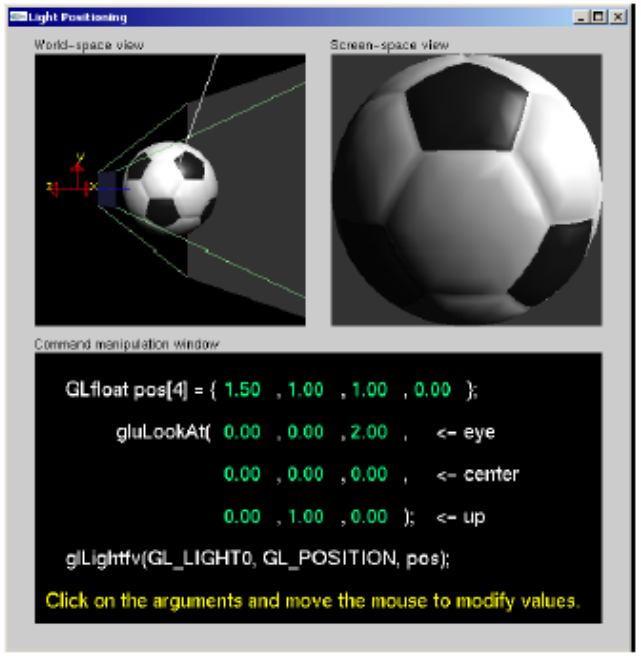
```
glEnable(GL_LIGHTING);
```

- aktiviranje izvora svetlosti

```
glEnable(GL_LIGHT0);
```

Primer korišćenja svetla

Tutorial: lightposition.exe



Uočiti razliku između pozicionog i direkcionog svetla.

Definisanje karakteristika materijala

void **glMaterial**{if}[v](GLenum *face*, GLenum *pname*, *TYPE param*);

Parameter ***face*** može biti:

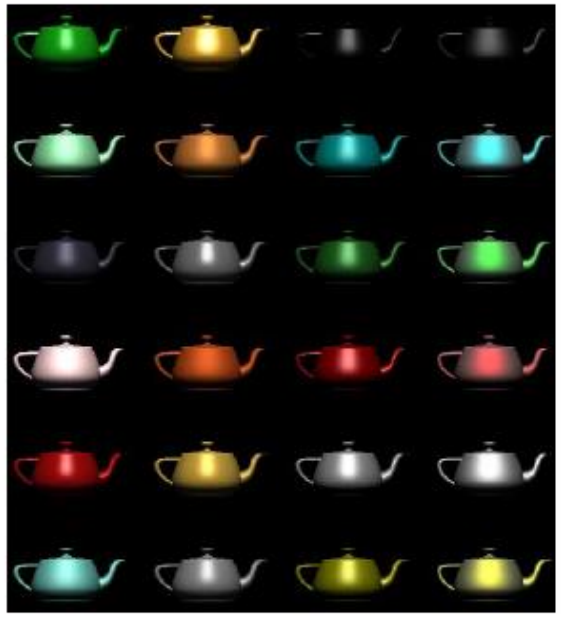
GL_FRONT,

GL_BACK, ili

GL_FRONT_AND_BACK

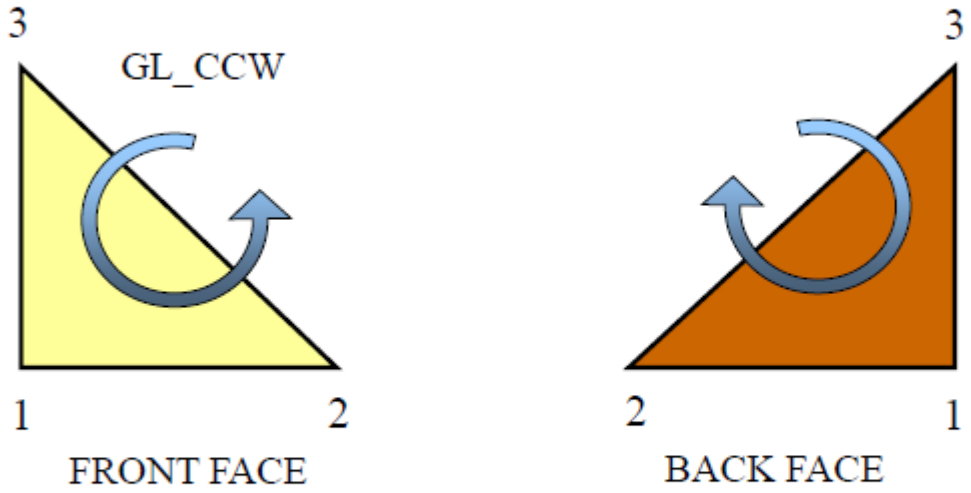
i određuje stranu (lice) objekta na koju se materijal primenjuje.

Pname je svojstvo materijala koje se menja, a ***param*** vrednost na koju se postavlja.



Orijentacija poligona

Prednja i zadnja strana određuje se na osnovu redosleda zadavanja temena. Po *default*-u smer zadavanja temena suprotan od kazaljke na satu definiše prednju stranu (FRONT FACE).



Podrazumevani način definisanja prednje strane može se promeniti pozivom funkcije:

void glFrontFace(GLenum *mode* **);**

gde *mode* može imati vrednost **GL_CW** ili **GL_CCW.**

Parametri materijala i njihove vrednosti

Parametar	Podrazumevana vrednost	Značenje
GL_AMBIENT	(0.2, 0.2, 0.2, 1.0)	Ambijentalna boja materijala
GL_DIFFUSE	(0.8, 0.8, 0.8, 1.0)	Difuziona boja materijala
GL_AMBIENT_AND_DIFFUSE	-	Ambijentalna i difuziona boja materijala
GL_SPECULAR	(0.0, 0.0, 0.0, 1.0)	Odsjaj materijala
GL_SHININESS	0.0	Eksponet odsjaja
GL_EMISSION	(0.0, 0.0, 0.0, 1.0)	Emisiona boja materijala
GL_COLOR_INDEXES	(0,1,1)	Kolor indeksi

Na providnost materijala (*alpha* vrednost) utiče samo *alpha* kanal difuzione komponente materijala.

Parametri materijala i njihove vrednosti

- **difuziona i ambijentalna refleksija** — najviše utiču na izgled materijala i obzirom da su obično se postavljaju na istu vrednost (zato se mogu istovremeno definisati parametrom GL_AMBIENT_AND_DIFFUSE)

```
GLfloat mat_amb_diff[] = { 0.1, 0.5, 0.8, 1.0 };
glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT_AND_DIFFUSE, mat_amb_diff);
```

- **odsjaj** — utiče na naš doživljaj materijala (metalne i plastične objekte vidimo kao sjajne, dok talk ili kreda uopšte nemaju odsjaj), a definišu ga:
  - GL_SPECULAR – def. boju odsjaja
  - GL_SHININESS – def. intenzitet odsjaja, kreće se u opsegu [0.0, 128.0], pri čemu veća vrednost znači površinski manji ali svetliji odsjaj (veća fokusiranost)

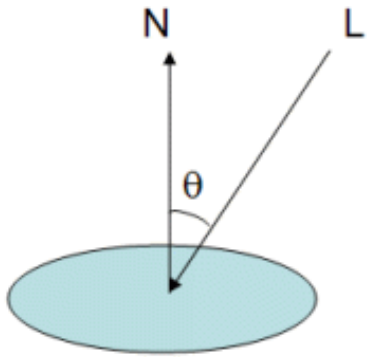
```
GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat low_shininess[] = { 5.0 };
glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
glMaterialfv(GL_FRONT, GL_SHININESS, low_shininess);
```

- **emisija** — def. RGBA boju emitovane svetlosti (koja ne utiče na osvetljenje scene). Nema smisla za većinu objekata, osim ako oni ne predstavljaju izvore svetlosti (sijalice, lampe, zvezde, ...)

```
GLfloat mat_emission[] = {0.3, 0.2, 0.2, 0.0};
glMaterialfv(GL_FRONT, GL_EMISSION, mat_emission);
```

material.c (exe)

Difuziona komponenta

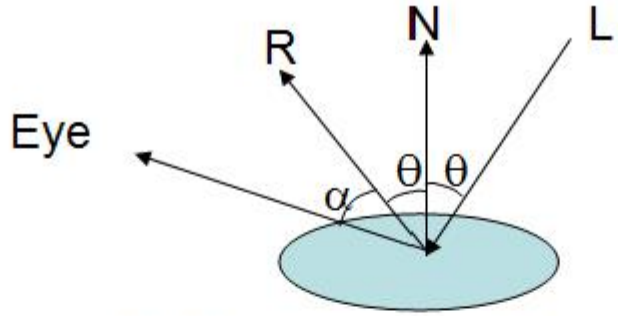


Difuziona komponenta ne zavisi od pozicije posmatrača, već samo od ugla upadnog svetlosnog zraka i normale na površinu.

Lambertov "kosinusni" zakon:

$$I_0 = L_d + M_d * \cos(\Theta)$$

Odsjaj – Phong metod



Odsjaj je proporcionalan kosinusu ugla reflektovanog zraka (R) i vektora posmatrača (Eye).

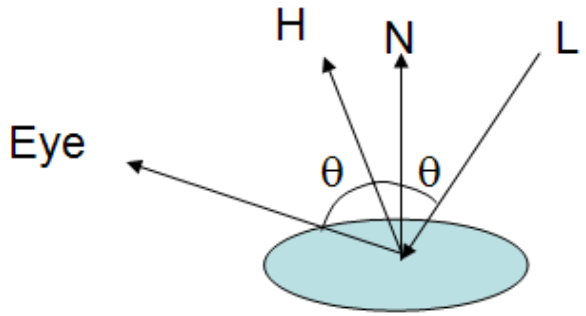
$$\mathbf{R = 2N\,(L \cdot N) - L}$$

$$\mathbf{Spec = (R \cdot Eye)^S * L_s * M_s}$$

s – sjaj, $s \in [0,128]$



Odsjaj – Blinn - Phong metod



Jednostavnija i brža varijanta računanja odsjaja je Blinn-Phong metoda. Odsjaj je proporcionalan kosinusu ugla *half* vektora (H) i vektora posmatrača (Eye). Half vektor simetrale ugla koga obrazuju padni svetlosni zrak i vektor posmatrača.

$$\mathbf{H = L + Eye}$$

$$\mathbf{Spec = (N \cdot H)^S * L_s * M_s}$$

Efikasno menjanje samo jedne karakteristike materijala

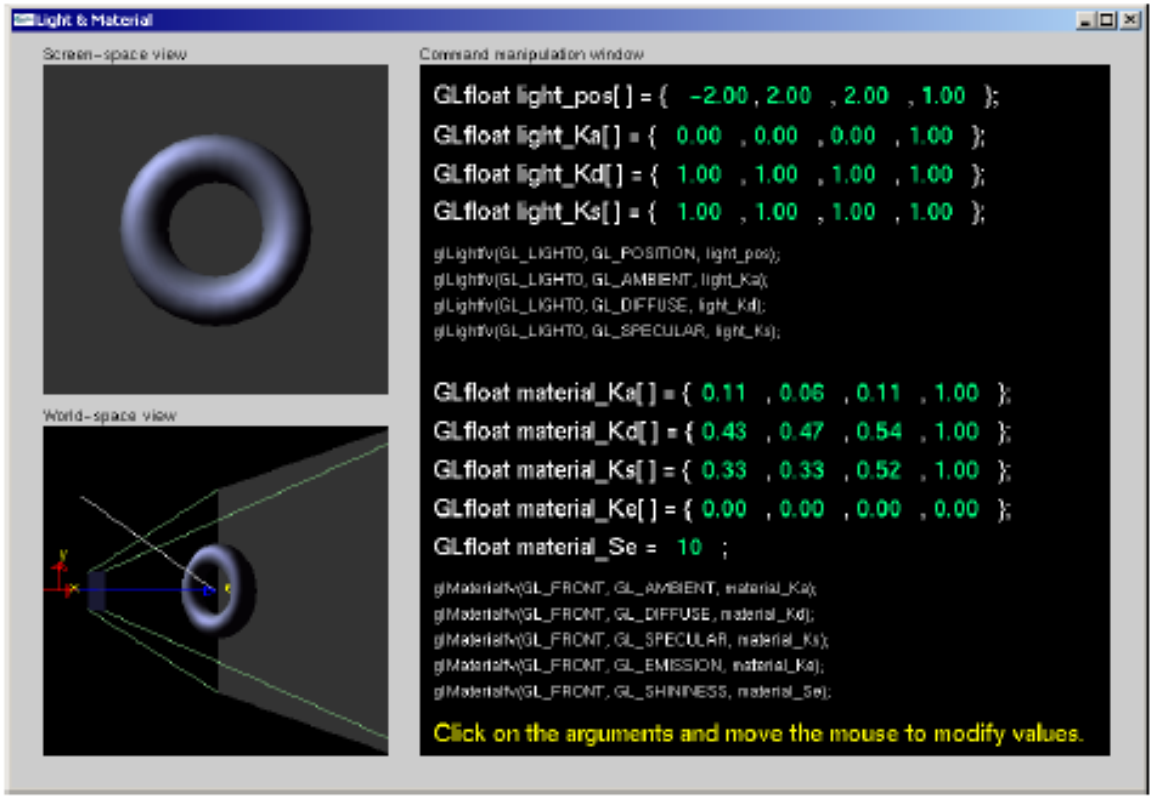
void **glColorMaterial**(GLenum *face*, GLenum *mode*);

Vezuje određenu karakteristiku materijala za boju koja se postavlja funkcijom **glColor**. Da bi imala dejstvo mora se pozvati **glEnable(GL_COLOR_MATERIAL)** nakon **glColorMaterial**.

```
glColorMaterial(GL_FRONT, GL_DIFFUSE);
glEnable(GL_COLOR_MATERIAL);
glColor3f(0.2, 0.5, 0.8);
/* draw some objects here */
glColor3f(0.9, 0.0, 0.2);
/* draw other objects here */
glDisable(GL_COLOR_MATERIAL);
```

Primer korišćenja materijala

Tutorial: lightmaterial.exe



Dodatni primer: colormat.c (exe)

Matematika osvetljenja

$$boja_temena = emission_{material} +$$
$$ambient_{light_model} * ambient_{material} +$$
$$\sum_{i=0}^{n-1} \left(\frac{1}{k_c + k_l d + k_q d^2} \right)_i * (spotlight_effect)_i *$$
$$[\ ambient_{light} * ambient_{material} +$$
$$(\max\{L \cdot n, 0\}) * diffuse_{light} * diffuse_{material} +$$
$$(\max\{s \cdot n, 0\})^{shininess} * specular_{light} * specular_{material}\ ]_i$$

L - Jedinični vektor usmeren od temena ka izvoru svetlosti

n – Vektor normale

s – Normalizovani zbir dva jedinična vektora (1. teme-izvor svetlosti, 2. teme-posmatrač) – "half" vektor

Hvala na pažnji!