Laborator 8
Iluminarea

Redarea luminilor în OpenGl este o sarcinŁ care implicŁ setari în mai multe direcţii:
ÅsetŁri generale ale mediului OpenGL ki ale înterpretŁrii luminii pe diferite feţe;
Åsurse de lumini;
ÅpropietŁţile surselor de lumini;
Åmaterial;
Ånormale la suprafaţŁ;

Interpretarea modelului luminii naturale în OpenGL

Pentru început sŁ ne imaginŁm o scena complexa din lumea reala. Ea este formata din obiecte cu diferite forme si texturi, dar si din multe lumini,reflexii de lumîni sau umbre. Din punct de vedere fizic, luminŁ este o energie fotonica care se misca prin spatiu aproape înstantaneu. Cand un obiect primeste o raza de luminŁ, el pastreaza o parte din energie pentru el, iar restul o imprastie în mediu. Partea care o primeste pentru el se transforma în caldura, iar partea care o emana va determîna practice culoarea obiectului.
În OpenGL, se considera ca într-o scena oarecare pot apare mai multe tipuri de lumîni de felul urmator:
1. luminŁ ambientala
2. luminŁ difuza
3. luminŁ speculara
4. luminŁ emisa

Lumină ambientală

Aceasta este lumina care este prezentŁ în atmosferŁ ñpur ki simpluò. Nu putem spune despre ea nici cŁ vine dintr-o anumitŁ sursŁ, nici cŁ este stralucitoare , nici cŁ se reflecta de oglindŁ. Este provenitŁ de la alte surse de lumini în urma reflexilor repetate, nu se mai poate stabili exact unde a fost sursa ki cum a ajuns lumina în locul respectiv.
Un exemplu bun de luminŁ ambientalŁ este cea vazutŁ într-un hol care primekte luminŁ numai de la sursele din camerele vecine. Lumina venitŁ poate fi amestecatŁ din razele de luminŁ venite din numeroase reflexii.
Doar luminŁ ambientalŁ nu ne ajutŁ sŁ vedem formele obiectelor colorate la fel. Motivul? Energia luminoasŁ este identicŁ pt. fiecare faŞŁ (nu depinde de orientare) si prin urmare culoarea vazutŁ este exact aceiaki.

Lumină Difuză

Aceasta este lumina provenitŁ de la o sursŁ anume. Ea este formatŁ din mai multe raze care se imprŁktie în toate direcŞile. Acum lumina obiectului este ki calculatŁ în funcŞie de direcŞia razei care lumineazŁ respectivul obiect. Sursa de luminŁ poate fi la înfinit (imaginaŞi-vŁ luminŁ solarŁ într-o zi seninŁ) sau poate fi apropiatŁ de obiect (imaginaŞi-vŁ un bec).
DiferenŞa dintre cele doua tipuri de surse se observŁ ki în direcŞia razelor.La soare, razele vin paralele, la bec razele chiar se vŁd cum emanŁ dintr-un punct fix. LuminŁ difuzŁ se apropie cel mai mult de majoritatea luminilor din lumea realŁ, reprezintŁ clasa luminilor cu sursŁ ki direcţie. De reŞinut cŁ poziŞia observatorului nu înfluenŞeazŁ culoarea obiectului în acest caz, cu alte cuvinte culoarea este transmisŁ la fel în toate direcŞile.

Lumina specular

AceastŁ este componenta care dŁ stralucire obiectelor. Se aseamanŁ cu lumina difuzŁ, însa singura diferenŞŁ este cŁ speculara este focalizatŁ.
Pentru a înŢelege conceptul e bine sŁ ne imaginŁm o raza laser, sau un reflector de foarte bunŁ calitate. Razele care pleacŁ din reflector trebuie sŁ fie paralele. Obiectele resping aceastŁ luminŁ tot într-un mod specular si prin urmare aceastŁ luminŁ nu poate fi vazutŁ din orice poziŞie. Ochiul trebuie sa se afle chiar pe direcŞia ei.

Lumină emisă

Aceasta este lumina pe care un obiect o emite prin el însuki. În OpenGL aceastŁ nu mai devine ki sursŁ pentru alte obiecte din jurul lui. De regulŁ nu este nevoie sŁ fie folositŁ decât dacŁ se dorekte obŞinerea de obiecte incandescente (lŁmpi, focuri, etc..)

Culorile luminii şi materialelor în OpenGL

De remarcat cŁ lumina are ki culoare. ExistŁ în lumea reala si lumini verzi si lumini albastre, dar ki lumini obiknuite albe. De cele mai multe ori, obiectul amestecŁ culorile primite de la luminŁ cu culorile sale naturale,putând rezulta chiar o culoare diferitŁ de culoarea iniŞialŁ a obiectului.
În OpenGL culoarea poate fi specificatŁ prin cele 3 componente RGB.
Acestea sunt nikte valori reale între 0 ki 1. De exemplu o luminŁ cu (R,G,B)=(1,1,1) reprezintŁ o luminŁ albŁ, cea mai puternicŁ. O luminŁ (0,0,1) reprezintŁ o luminŁ albastruie. În openGL atât luminŁ cât ki corpul au setate aceste valori pentru fiecare tip de luminŁ (ambientalŁ, difuzŁ ki specularŁ). LuminŁ reflectatŁ de un anumit obiect se obŞine înmulŞind valoarea culorii cu valoarea luminii ï bineînŞeles ambele trebuie sŁ se refere la acelaki tip de luminŁ.

8.1. Specificarea culorilor pentru lumină şi materiale

FuncŞile **glColor3** specificŁ un triplet (R, G, B). FuncŞile **glColor4** adaugŁ tripletului (R,G,B) o valoare de **opacitate**, numitŁ **valoarea alfa (A)**.

OpenGL permite utilizarea a douŁ moduri de reprezentare a culorilor de afikare:

- **modul RGBA**: pentru fiecare pixel se memoreazŁ valorile R, G, B ki A.
- **modul indexat**: pentru fiecare pixel se memoreazŁ un numŁr, reprezentând un index într-o tabela de culori.

Valorile R, G, B ki A variazŁ în domeniul [0, 1].

În modul RGBA, pentru selectarea culorii curente de desenare se folosesc funcŞile **glColor***.

```
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);
```

Valoarea implicitŁ a opacitŁŞi este 1.0.

Pentru versiunile funcŞiei **glColor*** care acceptŁ valori de tip real, domeniul acestora trebuie sŁ fie în intervalul [0, 1]. Valorile din afara intervalului [0,1] sunt trunchiate la valori în

intervalul [0,1] când sunt folosite ca parametri direcŞ, dar nu sunt trunchiate dacŁ sunt folosite pentru a modifica parametrii de material ki iluminare.

Valorile parametrilor de culoare specificate prin numere întregi sunt convertite în valori reale, conform tabelului 8.1

	Tip	**Valoarea minimă**	**Valoarea minimă se mapeaza la**	**Valoarea maximă**	**Valoarea maximă se mapează la**
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

Tabelul 8.1 Conversia valorilor de culoare la numere reale

Componentele culorilor specificate prin funcŞile **glColor*** au semnificaŞi diferite dupŁ cum sunt folosite pentru lumisa emisŁ de o sursŁ sau lumina reflectatŁ de o suprafaŞ. Pentru o sursŁ de luminŁ, valorile componentelor R, G ki B corespund unui procent din intensitatea maximŁ pentru fiecare culoare. DacŁ valorile R, G ki B sunt 1.0 atunci lumina este alb strŁlucitor. DacŁ valorile sunt 0.5, culoarea este albŁ, iar la jumŁtate din intensitate apare gri. DacŁ R=G=1 ki B=0 lumina apare galbenŁ.

Pentru proprietŁŞile de material, valorile componentelor R, G ki B corespund proporŞilor reflectate din acea culoare. DacŁ R=1, G=0.5 ki B=0 pentru un material, atunci acel material va reflecta toatŁ lumina incidentŁ roKie, jumŁtate din lumina incidentŁ verde ki nimic din lumina incidentŁ albastrŁ. Cu alte cuvinte dacŁ o sursŁ de luminŁ are componentele (LR, LG, LB) ki un material are componentele corespunzŁtoare (MR, MG, MB) atunci ignorându-se toate celelalte efecte ale reflectivitŁŞi, lumina ce este perceputŁ de ochiul observatorului este datŁ de formula (LR*MR, LG*MG, LB*MB).

Analog dacŁ sunt douŁ surse de luminŁ, cu componentele (R1, G1, B1) ki (R2, G2, B2), atunci lumina rezultatŁ va fi datŁ de (R1+R2, G1+G2, B1+B2). DacŁ oricare dintre componentele rezultate sunt mai mari ca 1, atunci componenta respectivŁ va fi trunchiatŁ la 1.

8.2. Specificarea modelului de iluminare

Modelele de iluminare care pot fi folosite în OpenGL sunt **modelul Lambert** ki **modelul Gouraud**. Modelul de iluminare dorit se specificŁ cu ajutorul funcŞei **glShadeModel**.

```
void glShadeModel (GLenum mode);
```

- *mode* specifică modelul de iluminare ce va fi selectat; el poate avea valorile:
  - **GL_SMOOTH**: este valoarea implicită, prin care se selectează modelul Gouraud
  - **GL_FLAT** : selectează modelul Lambert.

În **modelul Gouraud** culoarea fiecărui vârf este tratată în mod individual. Pentru un segment de dreaptă culoarea se obţine prin interpolarea culorilor vârfului. Pentru un poligon, culorile punctelor interioare se obţin prin interpolare pe baza culorilor vârfurilor.

Exemplu : în fişierul exemplul urmator se afişează un triunghi folosind modelul de iluminare Gouraud.

```
/* fişierul exemplu */
#include "glut.h"

void init(void)
{
   glClearColor (0.0, 0.0, 0.0, 0.0);
   glShadeModel (GL_SMOOTH);
}

void triangle(void)
{
   glBegin (GL_TRIANGLES);
   glColor3f (1.0, 0.0, 0.0);
   glVertex2f (5.0, 5.0);
   glColor3f (0.0, 1.0, 0.0);
   glVertex2f (25.0, 5.0);
   glColor3f (0.0, 0.0, 1.0);
   glVertex2f (5.0, 25.0);
   glEnd();
}

void display(void)
{
   glClear (GL_COLOR_BUFFER_BIT);
   triangle ();
```

```
   glFlush ();
}

void reshape (int w, int h)
{
   glViewport (0, 0, (GLsizei) w, (GLsizei) h);
   glMatrixMode (GL_PROJECTION);
   glLoadIdentity ();
   if (w <= h)
      gluOrtho2D (0.0, 30.0, 0.0, 30.0*(GLfloat) h/(GLfloat) w);
   else
      gluOrtho2D (0.0, 30.0*(GLfloat) w/(GLfloat) h, 0.0, 30.0);
   glMatrixMode(GL_MODELVIEW);
}

int main(int argc, char** argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
   glutInitWindowSize (500, 500);
   glutInitWindowPosition (100, 100);
   glutCreateWindow (argv[0]);
   init ();
   glutDisplayFunc(display);
   glutReshapeFunc(reshape);
   glutMainLoop();
   return 0;
}
```

În **modelul Lambert** culoarea unei primitive este datŁ de culoarea unui singur vârf. Culoarea unui segment de dreaptŁ este datŁ de culoarea celui de-al doile vârf. Culoarea unui poligon este datŁ de culoarea unui vârf, conform tabelului 8.2. Vârfurile poligoanelor sunt numerotate începând cu 1. Pentru a evita confuzii în ceea ce priveşte culoarea de desenare în modelul Lambert se va specifica o singurŁ culoare pentru o primitivŁ.

Tipul poligonului	**Vârful folosit pentru selectarea culorii poligonului**
Poligon singular	1
triangle strip	i+2

triangle fan	i+2
independent triangle	3i
quad strip	2i+2
independent quad	4i

Tabelul 8.2. Selectarea culorii pentru un poligon în modelul Lambert

8.3. Iluminarea unei scene 3D

Pașii de adăugare a iluminării într-o scenă sunt următorii :

1) Definirea vectorilor normală pentru fiecare vârf al tuturor obiectelor. Aceste normale determină orientarea relativă a obiectelor față de sursele de lumină ;
2) Crearea, selectarea și poziționarea uneia sau a mai multor surse de lumină ;
3) Crearea și selectarea unui model de iluminare care definește nivelul luminii globale, ambiante și poziția observatorului ;
4) Definirea proprietăților de material pentru obiectele din scenă.

Exemplu

Programul din fișierul urmator realizează afișarea unei sfere cu iluminare folosind o sursă de lumină. Normalele pentru sferă sunt definite de funcția **glutSolidSphere.**

```
/* fişierul exemplulsfera */
#include "glut.h"

void init(void)
{
   GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
   GLfloat mat_shininess[] = { 50.0 };
   GLfloat light_position[] = { 1.0, 1.0, 1.0, 0.0 };
   glClearColor (0.0, 0.0, 0.0, 0.0);
   glShadeModel (GL_SMOOTH);

   glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
   glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);
   glLightfv(GL_LIGHT0, GL_POSITION, light_position);

   glEnable(GL_LIGHTING);
```

```
   glEnable(GL_LIGHT0);
   glEnable(GL_DEPTH_TEST);
}

void display(void)
{
   glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   glutSolidSphere (1.0, 20, 16);
   glFlush ();
}

void reshape (int w, int h)
{
   glViewport (0, 0, (GLsizei) w, (GLsizei) h);
   glMatrixMode (GL_PROJECTION);
   glLoadIdentity();
   if (w <= h)
      glOrtho (-1.5, 1.5, -1.5*(GLfloat)h/(GLfloat)w,
         1.5*(GLfloat)h/(GLfloat)w, -10.0, 10.0);
   else
      glOrtho (-1.5*(GLfloat)w/(GLfloat)h,
         1.5*(GLfloat)w/(GLfloat)h, -1.5, 1.5, -10.0, 10.0);
   glMatrixMode(GL_MODELVIEW);
   glLoadIdentity();
}

int main(int argc, char** argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
   glutInitWindowSize (500, 500);
   glutInitWindowPosition (100, 100);
   glutCreateWindow (argv[0]);
   init ();
   glutDisplayFunc(display);
   glutReshapeFunc(reshape);
   glutMainLoop();
   return 0;
}
```

8.3.1. Crearea, poziţionarea şi activarea uneia sau a mai multor surse de lumină

În programul din fişierul de mai sus se foloseşte o singură sursă de lumină albă. Poziţia sa este specificată prin apelul funcţiei **glLightfv**. Acest exemplu foloseşte culoarea implicită pentru sursa de lumina 0 (**GL_LIGHT0**), care este albă; dacă se doreşte o sursă de lumină având o altă culoare se va folosi funcţia **glLight*.** Într-o scenă pot fi incluse cel mult 8 surse de lumină diferite. Culoarea implicită a acestor surse de lumină este negru. Ele pot fi poziţionate la o distanţă finită sau infinită faţă de scenă. Sursele de lumină pot produce un fascicul de lumină mai îngust sau mai larg. După ce au fost definite caracteriticile surselor de lumină, acestea trebuie să fie activate folosind funcţia **glEnable.** De asemenea, trebuie apelată funcţia **glEnable** având ca parametru **GL_LIGHTING** pentru a activa efectuarea calculelor de iluminare (în mod implicit acestea sunt dezactivate).

Crearea surselor de lumină

Sursele de lumină au o serie de proprietăţi cum sunt culoarea, poziţia şi direcţia. Funcţia folosită pentru a specifica toate proprietăţile luminii este **glLight***.

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

Funcţia crează o sursă de lumină.

- *light* specifică sursa de lumină creată; poate avea una din următoarele valori: **GL_LIGHT0**, **GL_LIGHT1**, ... , sau **GL_LIGHT7**;
- *pname* specifică caracteristicile luminii, conform tabelului 8.3;
- *param* indică valorile caracteristicilor setate prin *pname.*

Valorile implicite ale parametrului *param* pentru valorile parametrului *pname*

pname	Valoarea implicita	Observatie
GL_AMBIENT	(0.0, 0.0, 0.0, 1.0)	Intensitatea ambiantă a luminii (RGBA)
GL_DIFFUSE	(1.0, 1.0, 1.0, 1.0)	Intensitatea luminii difuze (RGBA)
GL_SPECULAR	(1.0, 1.0, 1.0, 1.0)	Intensitatea luminii speculare (RGBA)
GL_POSITION	(0.0, 0.0, 1.0, 0.0)	Poziţia (*x, y, z, w*) a luminii
GL_SPOT_DIRECTION	(0.0, 0.0, -1.0)	Direcţia (*x, y, z*) spotului de lumină
GL_SPOT_EXPONENT	0.0	Exponentul spotului de lumină
GL_SPOT_CUTOFF	180.0	Unghiul spotului de lumină
GL_CONSTANT_ATTENUATION	1.0	Factor de atenuare constant

GL_LINEAR_ATTENUATION	0.0	Factor de atenuare liniar
GL_QUADRATIC_ATTENUATION	0.0	Factor de atenuare cuadric

Tabelul 8.3.

Valorile implicite din tabelul de mai sus pentru **GL_DIFFUSE** şi **GL_SPECULAR** se aplică doar pentru sursa de lumină **GL_LIGHT0**. Pentru celelate surse de lumină, valoarea implicită este (0.0, 0.0, 0.0, 1.0) atât pentru **GL_DIFFUSE** cât şi pentru **GL_SPECULAR**.

Exemplu : definirea culorilor, poziţiei şi a factorilor de atenuare pentru o sursă de lumină

```
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);
glLightf(GL_LIGHT0, GL_CONSTANT_ATTENUATION, 2.0);
glLightf(GL_LIGHT0, GL_LINEAR_ATTENUATION, 1.0);
glLightf(GL_LIGHT0, GL_QUADRATIC_ATTENUATION, 0.5);
```

O sursă de lumină poziţionată (care nu este plasată la infinit) poate acţiona ca un spot ï lumina va fi emisă sub forma unui con. Pentru a specifica unghiul dintre axele conului şi o rază de pe suprafaţa conului se foloseşte parametrul **GL_SPOT_CUTOFF**. Unghiul conului este de fapt dublul acestei valori după cum se arată în figura 8.1:

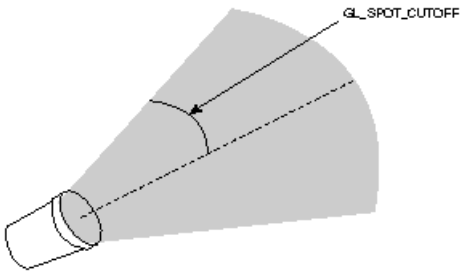

Figura 8.1

Valoarea implicită a parametrului **GL_SPOT_CUTOFF** este 180.0 (caracteristica de lumină spot este dezactivată), adică lumina este emisă în toate direcţiile. Valoarea parametrului **GL_SPOT_CUTOFF** trebuie să fie situată în intervalul [0.0,90.0] (altfel are valoarea 180.0).

Exemplu : setarea parametrului **GL_SPOT_CUTOFF** la 45^o

```
glLightf(GL_LIGHT0, GL_SPOT_CUTOFF, 45.0);
```

De asemenea, trebuie specificată o direcţie a spotului care determină axele conului de lumină:

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

Direcţia este specificată în coordonate obiect omogene. Valoarea implicită a direcţiei este (0.0, 0.0, -1.0).

Surse de lumină multiple

Într-o scenă pot fi definite cel mult 8 surse de lumină. Constatele folosite pentru referirea celor 8 surse de lumină sunt : **GL_LIGHT0**, **GL_LIGHT1**, **GL_LIGHT2**, **GL_LIGHT3**, é , **GL_LIGHT7**. Dacă se specifică o altă sursă de lumină trebuie să i se seteze parametrii corespunzători ca şi în cazul sursei de lumină **GL_LIGHT0**.

Exemplu : definirea unui spot de lumina albă:

```
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_LIGHT0, GL_SPOT_DIRECTION, spot_direction);
glLightf(GL_LIGHT1, GL_SPOT_EXPONENT, 2.0);

glEnable(GL_LIGHT1);
```

Observaţie: Pentru a activa fiecare sursă de lumină se foloseşte funcţia **glEnable** cu argumentul **GL_LIGHTING**. Pentru a dezactiva iluminarea se apelează functia **glDisable** cu acelaşi parametru.

8.3.2. Specificarea parametrilor modelului de iluminare

Funcţia **glLightModel*** defineşte parametrii modelului de iluminare:

```
void glLightModel{fi}(GLenum pname, TYPE param);
```

- *pname* reprezintă parametrul modelului de iluminare. El poate avea una din următoarele două valori:
  - **GL_LIGHT_MODEL_LOCAL_VIEWER,** caz în care *param* specifică modul de calcul al unghiului de reflexie.
  - **GL_LIGHT_MODEL_TWO_SIDE -** specifică feţele pentru care se face calculul de iluminare. Nu are nici un efect în calculele de lumină pentru puncte, linii şi bitmap-uri. Dacă param este 0 , calculele de iluminare se fac numai pentru feţele faţă şi vor fi folosite numai proprietăţile de material ale feţelor faţă. Altfel vor fi considerate atât feţele faţă cât şi feţele spate. În acest caz vârfurile feţelor spate vor fi luminate folosind proprietăţile de material ale feţelor spate (normalele vor fi inversate înainte de a calcula iluminarea).
- *param* reprezintă valoarea ce va fi folosită în corelaţie cu *pname*.

```
void glLightModel{fi}v( GLenum pname, const TYPE *params );
```

- *pname* **identifică** modelul de iluminare. El poate avea una din următoarele două valori:
  - **GL_LIGHT_MODEL_AMBIENT,** caz în care *params* va conţine componentele RGBA ale luminii ambiante.

- **GL_LIGHT_MODEL_LOCAL_VIEWER** are aceeaki semnificaŞie ca ki în cazul funcŞiei **glLightModel{fi};** în acest caz *param* devine *params*;
- **GL_LIGHT_MODEL_TWO_SIDE** are aceeaki semnificaŞie ca ki în cazul funcŞiei **glLightModel{fi};** în acest caz *param* devine *params*;

- *param* reprezintŁ valoarea ce va fi folositŁ în corelaŞie cu *pname*.

În exemplul din fikierul exemplul sfera singurul element care este definit explicit pentru modelul de iluminare este lumina ambiantŁ globalŁ. De asemenea, modelul de iluminare definekte ki poziŞia observatorului (la infinit sau la distanŞŁ finitŁ de scenŁ) precum ki modul de efectuare a calculelor de iluminare a feŞelor faŞŁ respectiv spate ale scenei. În programul din fikierul exemplul sfera se folosesc valorile implicite pentru acestea ï observatorul este plasat la infinit ki calculele de iluminare sunt efectuate pentru feŞele faŞŁ. Folosirea unui observator local scenei crekte complexitatea calculelor deoarece sistemul OpenGL trebuie sŁ calculeze unghiul dintre observator ki fiecare obiect. Folosind un observator plasat la infinit unghiul este ignorat ki rezultatele sunt ceva mai puŞin realiste.

În OpenGL modelul de iluminare are trei componente :

- Intensitatea luminii ambiante globale
- PoziŞionarea observatorului (local scenei sau la infinit)
- DiferenŞierea calculŁrii iluminŁrii pentru feŞele obiectelor faŞŁ, respectiv spate.

Lumina ambianta globală

Pentru a specifica intensitatea luminii ambiante globale ca RGBA se va folosi parametrul **GL_LIGHT_MODEL_AMBIENT** dupŁ cum urmeazŁ:

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

În exemplul de mai sus valorile folosite pentru *lmodel_ambient* sunt valorile implicite pentru **GL_LIGHT_MODEL_AMBIENT**.

Observator local sau plasat la infinit

PoziŞia observatorului afecteazŁ calculele pentru strŁlucirea produsŁ de lumina specularŁ, mai precis intensitatea strŁlucirii într-un vârf depinde de normala în acel vârf, direcŞia de la vârf la sursa de luminŁ ki direcŞia de la vârf la observator.

Cu un observator plasat la infinit direcŞia dintre observator ki orice vârf rŁmâne constantŁ. Un observator local va furniza rezultate mai aproape de realitate, dar direcŞia va trebui calculatŁ

pentru fiecare vârf, astfel cŁ în acest caz performaşele scad. În mod implicit observatorul este plasat la infinit.

Exemplu: definirea unui observator local în punctul de coordonate (0, 0, 0) (în coordonate observator):

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

Pentru a trece din nou la un observator plasat la infinit:

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

Calculul luminii pentru ambele feţe

În programul din fiкierul exemplul sfera numai feşele faşŁ sunt iluminate.

Exemplu: calcularea iluminŁrii pentru ambele feşe :

```
glLightModeli(LIGHT_MODEL_TWO_SIDE, GL_TRUE);
```

DacŁ se doreкte sŁ se calculeze apoi iluminarea numai pentru feşele faşŁ se va apela:

```
glLightModeli(LIGHT_MODEL_TWO_SIDE, GL_FALSE);
```

8.3.3. Definirea proprietăţilor de material pentru obiectele din scenă

ProprietŁşile de material ale unui obiect determinŁ modul în care el reflectŁ lumina. Pentru proprietŁşile unui material se pot specifica culorile ambiantŁ, difuzŁ, specularŁ, stralucirea sa кi culoarea oricŁrei lumini emise. Acestea pot fi setate folosind funcşa **glMaterialfv**.

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

FuncŞa specificŁ proprietŁşile de material folosite în calculul iluminŁrii.

- *face* reprezintă faţa obiectului pe care se vor aplica proprietăţile de material; poate avea una dintre valorile **GL_FRONT**, **GL_BACK** sau **GL_FRONT_AND_BACK**.
- *pname* indică proprietăţile de material;
- *param* indică una dintre valorile proprietăţilor selectate în parametrul *pname*.

Valorile posibile pentru *pname* sunt date în tabelul 8.3.

pname	Valoare implicita	Observaţie
GL_AMBIENT	(0.2, 0.2, 0.2, 1.0)	Culoarea ambiantă a materialului
GL_DIFFUSE	(0.8, 0.8, 0.8, 1.0)	Culoarea difuză a materialului
GL_AMBIENT_AND_DIFFUSE		Culoarea ambiantă şi difuză a materialului
GL_SPECULAR	(0.0, 0.0, 0.0, 1.0)	Culoarea specular a materialului
GL_SHININESS	0.0	Exponentul specular
GL_EMISSION	(0.0, 0.0, 0.0, 1.0)	Culoarea emisă a materialului
GL_COLOR_INDEXES	(0,1,1)	Indicii de culoare ambiantă, difuză şi specular

Tabelul 8.3.

Reflexia difuză şi ambiantă

Parametrii **GL_DIFFUSE** şi **GL_AMBIENT** setaţi cu funcţia **glMaterial*** afectează culoarea luminii ambiante şi difuze reflectate de un obiect.

Exemplu: asignarea simultană a aceleeaşi valori luminii reflectate difuze şi ambiante :

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

Reflexia specular

OpenGL permite setarea culorii RGBA a strălucirii speculare (folosind **GL_SPECULAR**) şi poate controla dimensiunea şi luminozitatea strălucirii (folosind **GL_SHININESS**). Parametrului **GL_SHININESS** i se poate asocia o valoare în domeniul [0.0, 128.0];cu cât valoarea este mai mare cu atât stălucirea este mai mică şi mai luminoasă.

Exemplu :

```
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);
```

Emisia

Prin asocierea unei culori RGBA parametrului **GL_EMISSION** se poate face ca un obiect sŁ parŁ cŁ furnizezŁ luminŁ de culoarea selectatŁ.

Exemplu:

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

Modificarea proprietăţilor de material

În fiкierul din primul exemplu toate vârfurile au asociate aceleaкi proprietŁŞi de material. Uneori se doreкte sŁ se asocieze vârfurilor aceluiaкi obiect proprietŁŞi de material diferite. De obiecei într-o scenŁ sunt mai multe obiecte кi fiecare au asociate proprietŁŞi de material diferite.

Exemplu : desenarea a opt sfere fiecare având alte proprietŁŞi de material :

```
GLfloat no_mat[] = { 0.0, 0.0, 0.0, 1.0 };
GLfloat mat_ambient[] = { 0.7, 0.7, 0.7, 1.0 };
GLfloat mat_ambient_color[] = { 0.8, 0.8, 0.2, 1.0 };
GLfloat mat_diffuse[] = { 0.1, 0.5, 0.8, 1.0 };
GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat no_shininess[] = { 0.0 };
GLfloat low_shininess[] = { 5.0 };
GLfloat high_shininess[] = { 100.0 };
GLfloat mat_emission[] = {0.3, 0.2, 0.2, 0.0};

glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

/* reflexie difuză */
glPushMatrix();
    glTranslatef (-3.75, 3.0, 0.0);
    glMaterialfv(GL_FRONT, GL_AMBIENT, no_mat);
    glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
    glMaterialfv(GL_FRONT, GL_SPECULAR, no_mat);
    glMaterialfv(GL_FRONT, GL_SHININESS, no_shininess);
    glMaterialfv(GL_FRONT, GL_EMISSION, no_mat);
```

```
    glutSolidSphere();
glPopMatrix();

/* reflexie difuză şi specular; strălucire redusă */
glPushMatrix();
    glTranslatef (-1.25, 3.0, 0.0);
    glMaterialfv(GL_FRONT, GL_AMBIENT, no_mat);
    glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
    glMaterialfv(GL_FRONT, GL_SHININESS, low_shininess);
    glMaterialfv(GL_FRONT, GL_EMISSION, no_mat);
    glutSolidSphere();
glPopMatrix();

/* reflexie difuză şi specular; stralucire ridicată */
glPushMatrix();
    glTranslatef (1.25, 3.0, 0.0);
    glMaterialfv(GL_FRONT, GL_AMBIENT, no_mat);
    glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
    glMaterialfv(GL_FRONT, GL_SHININESS, high_shininess);
    glMaterialfv(GL_FRONT, GL_EMISSION, no_mat);
    glutSolidSphere();
glPopMatrix();

/* reflexie difuză, emisie */
glPushMatrix();
    glTranslatef (3.75, 3.0, 0.0);
    glMaterialfv(GL_FRONT, GL_AMBIENT, no_mat);
    glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
    glMaterialfv(GL_FRONT, GL_SPECULAR, no_mat);
    glMaterialfv(GL_FRONT, GL_SHININESS, no_shininess);
    glMaterialfv(GL_FRONT, GL_EMISSION, mat_emission);
    glutSolidSphere();
glPopMatrix();
```

FuncŞa **glMaterialfv** este apelatŁ în mod repetat pentru a seta proprietŁŞile de material pentru fiecare sferŁ. Ea este reapelatŁ doar pentru acele proprietŁŞi care se modificŁ de la o sferŁ la alta (adicŁ de la un obiect la altul). Deoarece funcŞa **glMaterial*** are un cost de execuŞe mare este bine sŁ se minimizeze modificŁrile proprietŁŞilor de material. O altŁ tehnicŁ de a minimiza

costurile asociate cu modificarile proprietŁŞilor de material este folosirea funcŞiei **glColorMaterial :**

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

ProprietŁŞile de material specificate de parametrul *mode* ale feŞei specificate de parametrul *face* ia întotdeauna valoarea culorii curente. Astfel, o modificare asupra culorii curente (folosind funcŞia **glColor***) actualizeazŁ imediat proprietŁŞile de material specificate.

- *face* poate avea una dintre urmŁtoarele valori: **GL_FRONT**, **GL_BACK** sau **GL_FRONT_AND_BACK** (valoare implicitŁ);
- *mode* poate avea una din urmŁtoarele valori: **GL_AMBIENT**, **GL_DIFFUSE**, **GL_AMBIENT_AND_DIFFUSE** (valoare implicitŁ), **GL_SPECULAR** sau **GL_EMISSION**.

Observaţie: glColorMaterial actualizeazŁ proprietatea/proprietŁŞile de material specificate de parametrul ***mode*** ale faŞei/feŞelor specificate de parametrul *face*.

DupŁ apelul funcŞiei **glColorMaterial** trebuie apelatŁ funcŞia **glEnable** având ca parametru **GL_COLOR_MATERIAL**. Apoi culoarea curentŁ poate fi modificatŁ folosind funcŞia **glColor*** (sau alte proprietŁŞ de material folosind funcŞia **glMaterial***).

Exemplu:

```
glColorMaterial(GL_FRONT, GL_DIFFUSE);
glEnable(GL_COLOR_MATERIAL);
glColor3f(0.2, 0.5, 0.8);
/* afişează obiecte */
glColor3f(0.9, 0.0, 0.2);
/* afişează alte obiecte */
glDisable(GL_COLOR_MATERIAL);
```

FuncŞia **glColorMaterial** se va folosi de câte ori se dorekte modificarea unui singur parametru de material pentru majoritatea vârfurilor din scenŁ. DacŁ se dorekte modificarea mai multor parametri de material se va folosi funcŞia **glMaterial***.

Exemplu:Sfera solida iluminata

```
#include <GL\glut.h>

GLfloat xRotated, yRotated, zRotated;
GLdouble radius=0.5;
GLfloat qaBlack[] = {0.0, 0.0, 0.0, 1.0}; //Culoare Neagra
```

```
GLfloat qaGreen[] = {0.0, 1.0, 0.0, 1.0}; //Culoare Verde
GLfloat qaWhite[] = {1.0, 1.0, 1.0, 1.0}; //Culoare alba
GLfloat qaRed[] = {1.0, 0.0, 0.0, 1.0}; //Culoare alba

    // Setarea intensitatii de iluminare si culoarea
GLfloat qaAmbientLight[]    = {0.2, 0.2, 0.2, 1.0};
GLfloat qaDiffuseLight[]    = {0.8, 0.8, 0.8, 1.0};
GLfloat qaSpecularLight[]    = {1.0, 1.0, 1.0, 1.0};
GLfloat emitLight[] = {0.9, 0.9, 0.9, 0.01};
GLfloat Noemit[] = {0.0, 0.0, 0.0, 1.0};
    // Pozitionarea sursei de lumina
GLfloat qaLightPosition[]    = {0.5, 0, -3.5, 0.5};

void display(void);
void reshape(int x, int y);

void idleFunc(void)
{

     zRotated += 0.02;

    display();
}
void initLighting()
{

    // Activare iluminare
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);

     // Setarea intensitatii de iluminare si culoarea
       glLightfv(GL_LIGHT0, GL_AMBIENT, qaAmbientLight);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, qaDiffuseLight);
     glLightfv(GL_LIGHT0, GL_SPECULAR, qaSpecularLight);

    // Setarea pozitiei luminii
     glLightfv(GL_LIGHT0, GL_POSITION, qaLightPosition);

}

int main (int argc, char **argv)
{
    glutInit(&argc, argv);
     glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH );
    glutInitWindowSize(350,350);
    glutCreateWindow("Sfera Solida");
    initLighting();
    xRotated = yRotated = zRotated = 0.0;

    glutIdleFunc(idleFunc);
    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutMainLoop();
    return 0;
}

void display(void)
{
```

```
    glMatrixMode(GL_MODELVIEW);
    // sterge bufferul de desenare.
    glClear(GL_COLOR_BUFFER_BIT);
    // sterge matricea de identitate.
    glLoadIdentity();







    // translateaza desenul dupa z = -4.0
        glTranslatef(0.0,0.0,-5.0);


    // Transformarea de scalare
    glScalef(1.0,1.0,1.0);
    glRotatef(zRotated,0.0,0.0,1.0);
    // Seteaza proprietatile materialului
       glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, qaGreen);

    glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, qaGreen);

    glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, qaWhite);

    glMaterialf(GL_FRONT_AND_BACK, GL_SHININESS, 20);

    // deseneaza o sfera.
    glutSolidSphere(radius,25,25);
    // Curatare ecran

    glFlush();
    glutSwapBuffers();
    // Activare tampon dublu
   // glutSwapBuffers();
}

void reshape(int x, int y)
{
    if (y == 0 || x == 0) return;
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

    gluPerspective(39.0,(GLdouble)x/(GLdouble)y,0.6,21.0);
    glMatrixMode(GL_MODELVIEW);
    glViewport(0,0,x,y);  //Utilizati intreaga fereastra de randare
}
```

Exemplu: Sfera solida iluminata specular

```
#include <GL\glut.h>
#include <math.h>      // Pentru rutinele matematice (sqrt & trig).
GLfloat xRotated, yRotated, zRotated;
GLdouble radius=2;
```

```
GLfloat qaBlack[] = {0.0, 0.0, 0.0, 1.0}; // culoarea negru
GLfloat qaGreen[] = {0.0, 1.0, 0.0, 1.0}; // culoarea verde
GLfloat qaWhite[] = {1.0, 1.0, 1.0, 1.0}; // culoarea alb
GLfloat qaRed[] = {1.0, 0.0, 0.0, 1.0}; // culoarea alb

    // Seteaza intensitatea luminii si culoarea
GLfloat qaAmbientLight[]    = {0.1, 0.1, 0.1, 1.0};
GLfloat qaDiffuseLight[]    = {1, 1, 1, 1.0};
GLfloat qaSpecularLight[]    = {1.0, 1.0, 1.0, 1.0};
GLfloat emitLight[] = {0.9, 0.9, 0.9, 0.01};
GLfloat Noemit[] = {0.0, 0.0, 0.0, 1.0};
    // Poyitia sursei de lumina
GLfloat qaLightPosition[]    = {0, 0, 0, 1};

void display(void);
void reshape(int x, int y);

void idleFunc(void)
{
        if ( zRotated > 360.0 ) {
         zRotated -= 360.0*floor(zRotated/360.0);   // Nu permite intersectarea fluxului
      }

          if ( yRotated > 360.0 ) {
         yRotated -= 360.0*floor(yRotated/360.0);   // Nu permite intersectarea fluxului
      }
     zRotated += 0.01;
     yRotated +=0.01;

    display();
}
void initLighting()
{

    // Activarea iluminarii
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);

     // Seteaya intensitatea luminii si culoarea
       glLightfv(GL_LIGHT0, GL_AMBIENT, qaAmbientLight);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, qaDiffuseLight);
    glLightfv(GL_LIGHT0, GL_SPECULAR, qaSpecularLight);


}

int main (int argc, char **argv)
{
    glutInit(&argc, argv);
     glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH );
    glutInitWindowSize(350,350);
    glutCreateWindow("Sfera Solida");
    initLighting();

    xRotated = yRotated = zRotated = 0.0;

    glutIdleFunc(idleFunc);
    glutDisplayFunc(display);
```

```
    glutReshapeFunc(reshape);
    glutMainLoop();
    return 0;
}

void display(void)
{


    glMatrixMode(GL_MODELVIEW);
    // sterge bufferul de desenare.
    glClear(GL_COLOR_BUFFER_BIT);
    // sterge matrice de identitate.
    glLoadIdentity();

    glTranslatef(0.0,0.0,-20.0);


      //
   glPushMatrix();


    glTranslatef(0.0,0.0,0);
    // Seteaza proprietatile materialului
       glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, qaGreen);

    glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, qaGreen);

    glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, qaWhite);

    glMaterialf(GL_FRONT_AND_BACK, GL_SHININESS, 0.2);
    glutSolidSphere(radius,25,25);

     glPopMatrix();



    glPushMatrix();


    glRotatef(yRotated,0.0,1.0,0.0);
    glTranslatef(5.0,0.0,0.0);

    // Seteaza pozitia luminii
     glLightfv(GL_LIGHT0, GL_POSITION, qaLightPosition);
     glMaterialfv(GL_FRONT_AND_BACK, GL_EMISSION, emitLight);   // Creaza o sfera
stralucitoare (emisiva)
     glutSolidSphere(radius/6,25,25);
     glMaterialfv(GL_FRONT_AND_BACK, GL_EMISSION, Noemit);

    glPopMatrix();



    glFlush();
    glutSwapBuffers();
```

```
}

void reshape(int x, int y)
{
    if (y == 0 || x == 0) return;
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

    gluPerspective(39.0,(GLdouble)x/(GLdouble)y,0.6,40.0);
    glMatrixMode(GL_MODELVIEW);
    glViewport(0,0,x,y);  //Utilizeaza intreaga fereastra de randare
}
```

Exemplu:Iluminare cub rotitor

```
#include <stdlib.h>

#ifdef __APPLE__
#include <OpenGL/OpenGL.h>
#include <GLUT/glut.h>
#else
#include <GL/glut.h>
#endif


//Asociere butoane de la tastatura
void handleKeypress(unsigned char key, int x, int y) {
	switch (key) {
		case 27: //tasta Escape
			exit(0);
	}
}

//Initializare randare 3D
void initRendering() {
	glEnable(GL_DEPTH_TEST);
	glEnable(GL_COLOR_MATERIAL);
	glEnable(GL_LIGHTING); //Activare lumina
	glEnable(GL_LIGHT0); //Activare lumina #0
	glEnable(GL_LIGHT1); //Activare lumina #1
	glEnable(GL_NORMALIZE);
	//glShadeModel(GL_SMOOTH); //Activare consistenta de material shade
}

//Apelata cand fereastra este dimensionata
void handleResize(int w, int h) {
	glViewport(0, 0, w, h);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluPerspective(45.0, (double)w / (double)h, 1.0, 200.0);
}

float _angle = -70.0f;
```

```
//Deseneaza scena 3D
void drawScene() {
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

	glMatrixMode(GL_MODELVIEW);
	glLoadIdentity();

	glTranslatef(0.0f, 0.0f, -8.0f);

	//Adauga lumina ambientala
	GLfloat ambientColor[] = {0.2f, 0.2f, 0.2f, 1.0f}; //Culoare (0.2, 0.2, 0.2)
	glLightModelfv(GL_LIGHT_MODEL_AMBIENT, ambientColor);

	//Adauga lumina de pozitie
	GLfloat lightColor0[] = {0.5f, 0.5f, 0.5f, 1.0f}; //Culoare (0.5, 0.5, 0.5)
	GLfloat lightPos0[] = {4.0f, 0.0f, 8.0f, 1.0f}; //Pozitionare la (4, 0, 8)
	glLightfv(GL_LIGHT0, GL_DIFFUSE, lightColor0);
	glLightfv(GL_LIGHT0, GL_POSITION, lightPos0);

	//Adauga lumina directa
	GLfloat lightColor1[] = {0.5f, 0.2f, 0.2f, 1.0f}; //Culoare (0.5, 0.2, 0.2)
	//priveste din directia (-1, 0.5, 0.5)
	GLfloat lightPos1[] = {-1.0f, 0.5f, 0.5f, 0.0f};
	glLightfv(GL_LIGHT1, GL_DIFFUSE, lightColor1);
	glLightfv(GL_LIGHT1, GL_POSITION, lightPos1);

	glRotatef(_angle, 0.0f, 1.0f, 0.0f);
	glColor3f(1.0f, 1.0f, 0.0f);
	glBegin(GL_QUADS);

	//Front
	glNormal3f(0.0f, 0.0f, 1.0f);
	//glNormal3f(-1.0f, 0.0f, 1.0f);
	glVertex3f(-1.5f, -1.0f, 1.5f);
	//glNormal3f(1.0f, 0.0f, 1.0f);
	glVertex3f(1.5f, -1.0f, 1.5f);
	//glNormal3f(1.0f, 0.0f, 1.0f);
	glVertex3f(1.5f, 1.0f, 1.5f);
	//glNormal3f(-1.0f, 0.0f, 1.0f);
	glVertex3f(-1.5f, 1.0f, 1.5f);

	//Right
	glNormal3f(1.0f, 0.0f, 0.0f);
	//glNormal3f(1.0f, 0.0f, -1.0f);
	glVertex3f(1.5f, -1.0f, -1.5f);
	//glNormal3f(1.0f, 0.0f, -1.0f);
	glVertex3f(1.5f, 1.0f, -1.5f);
	//glNormal3f(1.0f, 0.0f, 1.0f);
	glVertex3f(1.5f, 1.0f, 1.5f);
	//glNormal3f(1.0f, 0.0f, 1.0f);
	glVertex3f(1.5f, -1.0f, 1.5f);

	//Back
	glNormal3f(0.0f, 0.0f, -1.0f);
	//glNormal3f(-1.0f, 0.0f, -1.0f);
	glVertex3f(-1.5f, -1.0f, -1.5f);
	//glNormal3f(-1.0f, 0.0f, -1.0f);
	glVertex3f(-1.5f, 1.0f, -1.5f);
```

```
        //glNormal3f(1.0f, 0.0f, -1.0f);
        glVertex3f(1.5f, 1.0f, -1.5f);
        //glNormal3f(1.0f, 0.0f, -1.0f);
        glVertex3f(1.5f, -1.0f, -1.5f);

        //Left
        glNormal3f(-1.0f, 0.0f, 0.0f);
        //glNormal3f(-1.0f, 0.0f, -1.0f);
        glVertex3f(-1.5f, -1.0f, -1.5f);
        //glNormal3f(-1.0f, 0.0f, 1.0f);
        glVertex3f(-1.5f, -1.0f, 1.5f);
        //glNormal3f(-1.0f, 0.0f, 1.0f);
        glVertex3f(-1.5f, 1.0f, 1.5f);
        //glNormal3f(-1.0f, 0.0f, -1.0f);
        glVertex3f(-1.5f, 1.0f, -1.5f);

        glEnd();

        glutSwapBuffers();
}

void update(int value) {
        _angle += 1.5f;
        if (_angle > 360) {
                _angle -= 360;
        }

        glutPostRedisplay();
        glutTimerFunc(25, update, 0);
}

int main(int argc, char** argv) {
        //Initializare GLUT
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
        glutInitWindowSize(400, 400);

        //Creare fereastra
        glutCreateWindow("Iluminare");
        initRendering();

        //Seteaza functiile
        glutDisplayFunc(drawScene);
        glutKeyboardFunc(handleKeypress);
        glutReshapeFunc(handleResize);

        glutTimerFunc(25, update, 0); //Adauga timpul

        glutMainLoop();
        return 0;
}
```

Exemplu:Iluminare ambientala –ceainic

```
#include <iostream>
#include <stdlib.h>
```

```
#ifdef __APPLE__
#include <OpenGL/OpenGL.h>
#include <GLUT/glut.h>
#else
#include <GL/glut.h>
#endif

using namespace std;

//Apelare la actionarea unei taste
void handleKeypress(unsigned char key, int x, int y) {


    switch(key){
        case 'a':
            glDisable(GL_LIGHT0);
            break;
        case 's':
            glEnable(GL_LIGHT0);
            break;

        case 'd':
            glDisable(GL_LIGHT1);
            break;
        case 'f':
            glEnable(GL_LIGHT1);
            break;
        case 27: //tasta Escape
            exit(0);
        default:
            break;
    }
    glutPostRedisplay();
}

//Initializare randare 3D
void initRendering() {
    glEnable(GL_DEPTH_TEST);
    glEnable(GL_COLOR_MATERIAL);
    glEnable(GL_LIGHTING); //Activare iluminare
    glEnable(GL_LIGHT0); //Activare iluminare #0
    glEnable(GL_LIGHT1); //Activare iluminare #1
    glEnable(GL_NORMALIZE); //Normalizarea automata a normalelor la suprafata
    //glShadeModel(GL_SMOOTH); //Activare culoare neteda
}

//Apelare cand fereastra este redimensionata
void handleResize(int w, int h) {
    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45.0, (double)w / (double)h, 1.0, 200.0);
}

float _angle = -70.0f;

//Deseneaza scena 3D
void drawScene() {
```

```
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glTranslatef(0.0f, 0.0f, -8.0f);

    //Adaugare iluminare ambientala
    GLfloat ambientColor[] = {0.2f, 0.2f, 0.2f, 1.0f}; //Color (0.2, 0.2, 0.2)
    glLightModelfv(GL_LIGHT_MODEL_AMBIENT, ambientColor);

    //Adaugare iluminare de pozitie
    GLfloat diffuseLightColor0[] = {1.0f, 1.0f, 1.0f, 1.0f};
    GLfloat specularLightColor0[] = {1.0f, 1.0f, 1.0f, 1.0f};
    GLfloat lightPos0[] = {1.0f, 1.0f, 0.0f, 1.0f};
    //glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuseLightColor0);
    glLightfv(GL_LIGHT0, GL_SPECULAR, specularLightColor0);
    glLightfv(GL_LIGHT0, GL_POSITION, lightPos0);

    //Adaugare lumina directionata
    GLfloat lightColor1[] = {0.5f, 0.2f, 0.2f, 1.0f}; //Culoare (0.5, 0.2, 0.2)
    //Apare din directia (-1, 0.5, 0.5)
    GLfloat lightPos1[] = {-1.0f, 0.5f, 0.5f, 0.0f};
    glLightfv(GL_LIGHT1, GL_DIFFUSE, lightColor1);
    glLightfv(GL_LIGHT1, GL_POSITION, lightPos1);
    glRotatef(_angle, 0.0f, 1.0f, 0.0f);
    glColor3f(1.0f, 1.0f, 0.0f);

    glutSolidTeapot(2.0);
    glutSwapBuffers();
}

void update(int value) {
    _angle += 1.5f;
    if (_angle > 360) {
        _angle -= 360;
    }

    glutPostRedisplay();
    glutTimerFunc(25, update, 0);
}

int main(int argc, char** argv) {
    //Initializare GLUT
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize(400, 400);
    //Creare fereastra
    glutCreateWindow("Iluminare ambientala");
    initRendering();
    //Setam functiile
    glutDisplayFunc(drawScene);
    glutKeyboardFunc(handleKeypress);
    glutReshapeFunc(handleResize);
    glutTimerFunc(25, update, 0); //Adaugam timp

    glutMainLoop();
    return 0;
}
```

Exemplu:Iluminare speculara –ceainic

```
#include <GL\glut.h>
#include <math.h>      // Pentru rutinele matematice (cum ar fi sqrt & trig).

GLfloat xRotated, yRotated, zRotated;
GLdouble radius=2;
GLfloat qaBlack[] = {0.0, 0.0, 0.0, 1.0}; //Culoare Neagra
GLfloat qaGreen[] = {1.0, 0.0, 0.0, 1.0}; //Culoare Verde
GLfloat qaWhite[] = {1.0, 1.0, 1.0, 1.0}; //Culoare Alba
GLfloat qaRed[] = {1.0, 0.0, 0.0, 1.0}; //Culoare Red Color

    // Seteaza intensitatea luminii si Culoarea
GLfloat qaAmbientLight[]    = {0.1, 0.1, 0.1, 1.0};
GLfloat qaDiffuseLight[]    = {1, 1, 1, 1.0};
GLfloat qaSpecularLight[]    = {1.0, 1.0, 1.0, 1.0};
GLfloat emitLight[] = {0.9, 0.9, 0.9, 0.01};
GLfloat Noemit[] = {0.0, 0.0, 0.0, 1.0};
    // Seteaza pozitionarea sursei de lumina
GLfloat qaLightPosition[]    = {0, 0, 0, 1};// pozitionarea luminii
GLfloat qaLightDirection[]    = {1, 1, 1, 0};// lumina directionata
void display(void);
void reshape(int x, int y);

void idleFunc(void)
{
        if ( zRotated > 360.0 ) {
         zRotated -= 360.0*floor(zRotated/360.0);   // Nu permite rotirea peste fluxul de
lumina
      }

          if ( yRotated > 360.0 ) {
         yRotated -= 360.0*floor(yRotated/360.0);   // Nu permite rotirea peste fluxul de
lumina
      }
     zRotated += 0.1;
     yRotated +=0.1;

    display();
}
void initLighting()
{

    // Activeaza iluminarea
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);

     // Seteaza intensitatea iluminarii si culoarea
       glLightfv(GL_LIGHT0, GL_AMBIENT, qaAmbientLight);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, qaDiffuseLight);
     glLightfv(GL_LIGHT0, GL_POSITION, qaLightPosition);
     glLightfv(GL_LIGHT0, GL_SPECULAR, qaSpecularLight);
    ////////////////////////////////////////////////


}
```

```
void keyboard(unsigned char key, int x, int y)
{


     if (key == 'l' || key == 'L')
    {
          glEnable(GL_LIGHTING);
    }
    else if (key == 'd' || key == 'D')
    {
        glDisable(GL_LIGHTING);
    }

}


int main (int argc, char **argv)
{
    glutInit(&argc, argv);
     glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH );
    glutInitWindowSize(350,350);
    glutCreateWindow("Ceainic -");
    initLighting();

    xRotated = yRotated = zRotated = 0.0;

    glutIdleFunc(idleFunc);
    glutDisplayFunc(display);
    glutKeyboardFunc(keyboard); // Inregistreaza apelul de la tastatura
    glutReshapeFunc(reshape);
    glutMainLoop();
    return 0;
}

void display(void)
{


    glMatrixMode(GL_MODELVIEW);
    // sterge tamponul de desenare.
    glClear(GL_COLOR_BUFFER_BIT);
    // sterge matricea de identitate.
    glLoadIdentity();

    glTranslatef(0.0,0.0,-20.0);

   glPushMatrix();


    glTranslatef(0.0,0.0,0);
    // Seteaza proprietatile materialului
       glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, qaRed);

    glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, qaRed);

     glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, qaWhite);

     glMaterialf(GL_FRONT_AND_BACK, GL_SHININESS, 1);
```

```
          glutSolidTeapot(radius);
      glPopMatrix();



    glPushMatrix();
    glRotatef(yRotated,0.0,1.0,0.0);
    glTranslatef(5.0,0.0,0.0);
     glMaterialfv(GL_FRONT_AND_BACK, GL_EMISSION, emitLight);   // Construieste sfera
stralucitoare (emisiva)
     glutSolidSphere(radius/6,25,25);
     glMaterialfv(GL_FRONT_AND_BACK, GL_EMISSION, Noemit);
    glPopMatrix();

    glPushMatrix();
    glRotatef(-yRotated,0.0,1.0,0.0);
    glTranslatef(5.0,0.0,0.0);
      glLightfv(GL_LIGHT0, GL_POSITION, qaLightPosition);
    glPopMatrix();


    glFlush();
    glutSwapBuffers();

}

void reshape(int x, int y)
{
    if (y == 0 || x == 0) return;
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

    gluPerspective(39.0,(GLdouble)x/(GLdouble)y,0.6,40.0);
    glMatrixMode(GL_MODELVIEW);
    glViewport(0,0,x,y);  //Utilizati fereastra de randare
}
```