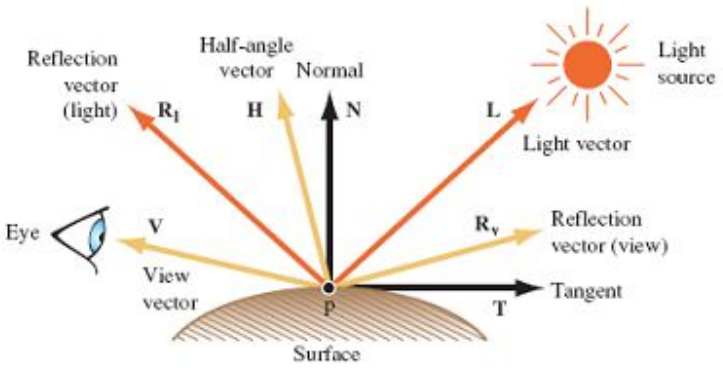
OPENGL (4 ლექცია)
განათება

OpenGL-ში გამოიყენება ფონგის განათების მოდელი.



განათების წყარო გამოასხივებს 3 ძირითად ფერს RGB(წითელი, მწვანე, ლურჯი). სცენის ობიექტის განათებაში თავისი წილი შეაქვთ შემდეგ კომპონენტებს:

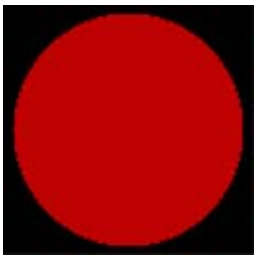
1. ესაა Ambient კომპონენტი:
ესაა გაფანტული შუქი რომელიც ყველა მხრიდან მოდის.

მარტო ამ კომპონენტით განათებული 3D ობიექტი გამოიყურება როგორც 2D და შეფერადებული.

2. Diffuse კომპონენტი:

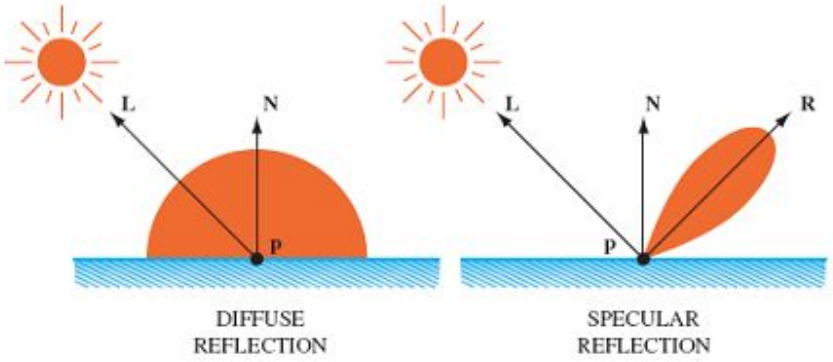
ამ დროს უკვე ითვალისწინება სინათლის წყაროს მიმართულება და გასანათებელი ობიექტი ემსავსება 3D-ს.

დაბლა სურათზე წარმოდგენელია სფერო:
ჯერ მარტო ამბიენტით და მერე დიფუზიური განათებით:

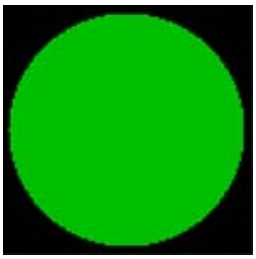



3. Specular კომპონენტი:

აქ სინათლის წყაროს მიმართულების გარდა გაითვალისწინება ობიექტის ზედაპირის გაპრიალებულობა და ხედვის ვექტორის კუთხეც. დიფუზიური კომპონენტისგან განსხვავებით(დიფუზიურის დროს თანამრად ხდება გაფანტვა ყველა მიმართულებით) აქ არეკვლისას დაცემის კუთხე არეკვლის კუთხის ტოლია. ანუ გაპრიალებული ზედაპირზე სპეკულარული სინათლის კომპონენტი მეტ წილს შეიტანს, და პირიქით მაგალითად ხალიჩის განათებაზე საერთოდ არ იმოქმედებს სპეკულარული კომპონენტი.



4. Emissive კომპონენტი



აქ განათების წყარო თვითონ ობიექტია. ანუ ობიექტი გამოსხივებს სინათლეს.
საბოლოო განათება ითვლება როგორ ამ 4 კომპონენტის ერთობლიობა.

$$L_{total} = L_{amb} + L_{spec} + L_{diff} + L_{em}$$

განათების მოდელის რეალიზაცია

1. პირველი რაც უნდა ვქნათ ჩვენ უნდა "ჩავრთოთ" განათება. ამით OpenGL-ს ვეუბნებით რომ, სცენის განათებას ჩვენ საკუთარ თავზე ვიღებთ.

glEnable(GL_LIGHTING);

2. უნდა დავაყენოთ განათების წყაროს თვისებები
glLightfv (ვექტორ პარამეტრებისთვის) glLightf(სკალარულ პარამეტრებისთვის) საშუალებით:

CODE

```
//ამბიენტ კომპონენტას მნიშვნელობები
GLfloat ambient[]  = { 0.3, 0.3, 0.3, 1.0 };
//დიფუზიური...
GLfloat diffuse[]  = { 1.0, 1.0, 1.0, 1.0 };
//სპეკულარული
GLfloat specular[]  = { 1.0, 1.0, 1.0, 1.0 };
//სინათლის წყაროს პოზიცია (x,y,z) კოორდინატები
GLfloat light_position[]  = { 0.0, 5.0, 2.0 };

glLightfv(GL_LIGHT0, GL_AMBIENT, ambient);
glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuse);
glLightfv(GL_LIGHT0, GL_SPECULAR, specular);
glLightfv(GL_LIGHT0, GL_POSITION, light_position);
```

სინათლის წყაროები გადანომრილია როგორც LIGHT0, LIGHT1...

3. გაჩუმებით ყველა წყარო გამორთულია. ამიტომ შემდეგი რაც უნდა ვქნათ ესაა უნდა ჩავრთოთ სინათლის წყარო. შეიძლება რამოდენიმე სინათლის წყაროც გვქონდეს.

glEnable(GL_LIGHT0);

ავანთეთ შუქი სცენაზე.

ეხლა საჭიროა კონკრეტულ ობიექტებს დავუნიშნოთ კონკრეტული მატერიალები: ქვემოთ მოცემულია რამოდენიმე მატერიალის პარამეტრების სია:

Material	GL_AMBIENT	GL_DIFFUSE	GL_SPECULAR	GL_SHININESS
Brass	0.329412 0.223529 0.027451 1.0	0.780392 0.568627 0.113725 1.0	0.992157 0.941176 0.807843 1.0	27.8974
Bronze	0.2125 0.1275 0.054 1.0	0.714 0.4284 0.18144 1.0	0.393548 0.271906 0.166721 1.0	25.6
Polished Bronze	0.25 0.148 0.06475 1.0	0.4 0.2368 0.1036 1.0	0.774597 0.458561 0.200621 1.0	76.8
Chrome	0.25 0.25 0.25 1.0	0.4 0.4 0.4 1.0	0.774597 0.774597 0.774597 1.0	76.8
Copper	0.19125 0.0735 0.0225 1.0	0.7038 0.27048 0.0828 1.0	0.256777 0.137622 0.086014 1.0	12.8
Polished Copper	0.2295 0.08825 0.0275 1.0	0.5508 0.2118 0.066 1.0	0.580594 0.223257 0.0695701 1.0	51.2
Gold	0.24725 0.1995 0.0745 1.0	0.75164 0.60648 0.22648 1.0	0.628281 0.555802 0.366065 1.0	51.2
Polished Gold	0.24725 0.2245 0.0645 1.0	0.34615 0.3143 0.0903 1.0	0.797357 0.723991 0.208006 1.0	83.2
Pewter	0.105882 0.058824 0.113725 1.0	0.427451 0.470588 0.541176 1.0	0.333333 0.333333 0.521569 1.0	9.84615

Material	GL_AMBIENT	GL_DIFFUSE	GL_SPECULAR	GL_SHININESS
Silver	0.19225 0.19225 0.19225 1.0	0.50754 0.50754 0.50754 1.0	0.508273 0.508273 0.508273 1.0	51.2
Polished Silver	0.23125 0.23125 0.23125 1.0	0.2775 0.2775 0.2775 1.0	0.773911 0.773911 0.773911 1.0	89.6
Emerald	0.0215 0.1745 0.0215 0.55	0.07568 0.61424 0.07568 0.55	0.633 0.727811 0.633 0.55	76.8
Jade	0.135 0.2225 0.1575 0.95	0.54 0.89 0.63 0.95	0.316228 0.316228 0.316228 0.95	12.8
Obsidian	0.05375 0.05 0.06625 0.82	0.18275 0.17 0.22525 0.82	0.332741 0.328634 0.346435 0.82	38.4
Pearl	0.25 0.20725 0.20725 0.922	1.0 0.829 0.829 0.922	0.296648 0.296648 0.296648 0.922	11.264
Ruby	0.1745 0.01175 0.01175 0.55	0.61424 0.04136 0.04136 0.55	0.727811 0.626959 0.626959 0.55	76.8
Turquoise	0.1 0.18725 0.1745 0.8	0.396 0.74151 0.69102 0.8	0.297254 0.30829 0.306678 0.8	12.8
Black Plastic	0.0 0.0 0.0 1.0	0.01 0.01 0.01 1.0	0.50 0.50 0.50 1.0	32
Black Rubber	0.02 0.02 0.02 1.0	0.01 0.01 0.01 1.0	0.4 0.4 0.4 1.0	10

ობიექტის დახატვამდე რომელსაც გვინდა რამე კონკრეტული მატერიალი დავუნიშნოთ, საჭიროა პარამეტრების დაყენება შემდეგნაერად. მაგალითად ავიღოთ ზემოთ მოყვანილი სიიდან:

Polished Copper

CODE

```
GL_AMBIENT (0.2295, 0.08825, 0.0275, 1.0)
GL_DIFFUSE (0.5508, 0.2118, 0.066, 1.0)
GL_SPECULAR (0.580594, 0.223257, 0.0695701, 1.0)
GL_SHININESS 51.2
```

შესაბამისი კოდი იქნება:

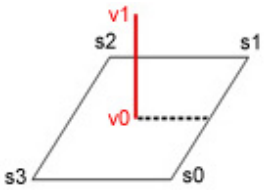
CODE

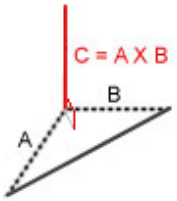
```
GLfloat mat_ambient[]  = { 0.2295, 0.08825, 0.0275, 1.0};
GLfloat mat_diffuse[]  = { 0.5508, 0.2118, 0.066, 1.0};
GLfloat mat_specular[]  = { 0.580594, 0.223257, 0.0695701, 1.0};
GLfloat mat_emission[]  = { 0.0, 0.0, 0.0, 1.0 };
GLfloat mat_Shininess = 51.2;

glMaterialfv(GL_FRONT, GL_AMBIENT,  mat_ambient);
glMaterialfv(GL_FRONT, GL_DIFFUSE,  mat_diffuse);
glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
glMaterialfv(GL_FRONT, GL_EMISSION, mat_emission);
glMaterialf(GL_FRONT,  GL_SHININESS, mat_Shininess);
```

ამის მერე დახატული ობიექტი იქნება შეფერადებული შესაბამისად.
გასათვალისწინებელია ერთი უმნიშვნელოვანესი რამე:

როგორ ვიცით OPENGL-ს განათების გამოსათვლელად ჭირდება ნორმალები. თუ ობიექტი "ხელითაა" გაკეთებული მას როგორ წესი ნორმალები არ აქვს და OPENGL არ აკეთებს ნორმალების ავტომატურ დათვლას. ასეთი ობიექტებისთვის მოგიწევთ თვითონ გამოთვალოთ ნორმალები, წინააღმდეგ შემთხვევაში განათება ან საერთო ან არაკორექტულად იმუშავებს

ზედაპირის ნორმალი როგორც ვიცით მიუთითებს იმას თუ საით "იყურება" ეს ზედაპირი.





ნორმალის გამოთვლა ადვილად ხდება როგორც 2 ვექტორის cross product-ი მესამე ვექტორი რომელის მიმართულია ამ ორის მართობულად. ამის შემდეგ უნდა მოხდეს ვექტორის ნორმალიზება:

ნორმალის გამოთვლა:

CODE

```
// This is how a vertex is specified in the base code
typedef struct vertex_s
{
   float x, y, z;
} vertex_t;

// normal(); - finds a normal vector and normalizes it
void normal (vertex_t v[3], vertex_t *normal)
{
   vertex_t a, b;

   // calculate the vectors A and B
   // note that v[3] is defined with counterclockwise winding in mind
   // a
   a.x = v[0].x - v[1].x;
   a.y = v[0].y - v[1].y;
   a.z = v[0].z - v[1].z;
   // b
   b.x = v[1].x - v[2].x;
   b.y = v[1].y - v[2].y;
   b.z = v[1].z - v[2].z;

   // calculate the cross product and place the resulting vector
   // into the address specified by vertex_t *normal
   normal->x = (a.y * b.z) - (a.z * b.y);
   normal->y = (a.z * b.x) - (a.x * b.z);
   normal->z = (a.x * b.y) - (a.y * b.x);

   // normalize
   normalize(normal);
}

void normalize (vector_t *v)
{
   // calculate the length of the vector
   float len = (float)(sqrt((v.x * v.x) + (v.y * v.y) + (v.z * v.z)));

   // avoid division by 0
   if (len == 0.0f)
       len = 1.0f;

   // reduce to unit size
   v.x /= len;
   v.y /= len;
   v.z /= len;
}
```

ნორმალის ინფორმაციის მიწოდება OPENGL-ში ხდება glNormal3f ფუნქციის საშუალებით (ან შერეული მასივების...), რომელიც უნდა დაყენდეს წვეროს ინფორმაციის შეყვანამდე glVertex3f.