ΔΗΜΟΚΡΙΤΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΡΑΚΗΣ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Εισαγωγή στα Γραφικά Υπολογιστών

Εργαστήριο 5

Σκηνή φωτισμού

- Δύο αντικείμενα Shader (με διαφορετικά αρχεία)

- Δύο κύβοι:

1) Αντικείμενο

2)Φως → μετασχηματισμοί:

```
glm::vec3 lightPos(1.2f, 1.0f, 2.0f);
model = glm::translate(model, lightPos);
model = glm::scale(model, glm::vec3(0.2f));
```

Σκηνή φωτισμού

- Vertex Shader (και του αντικειμένου και του φωτός):

```
#version 330
layout (location = 0) in vec3 in_position;
uniform mat4 model;
uniform mat4 view;
unifrom mat4 projection;
void main()
{
   gl_Position = projection * view * model * vec4(in_position, 1.0);
}
```

Σκηνή φωτισμού

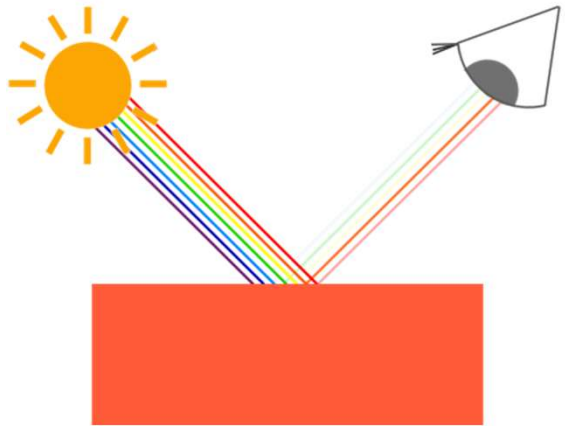
- Fragment Shader (του αντικειμένου):

```
#version 330 core
uniform vec3 objectColor;
uniform vec3 lightColor;

void main()
{
    gl_FragColor = vec4(lightColor * objectColor, 1.0);
}
```

Σκηνή φωτισμού

- Το τελικό χρώμα εξαρτάται από το χρώμα του αντικειμένου και το χώμα του φωτός:

$$(R_{eye}, G_{eye}, B_{eye}) = (R_{obj}, G_{obj}, B_{obj}) * (R_{light}, G_{light}, B_{ligjt})$$



Shader

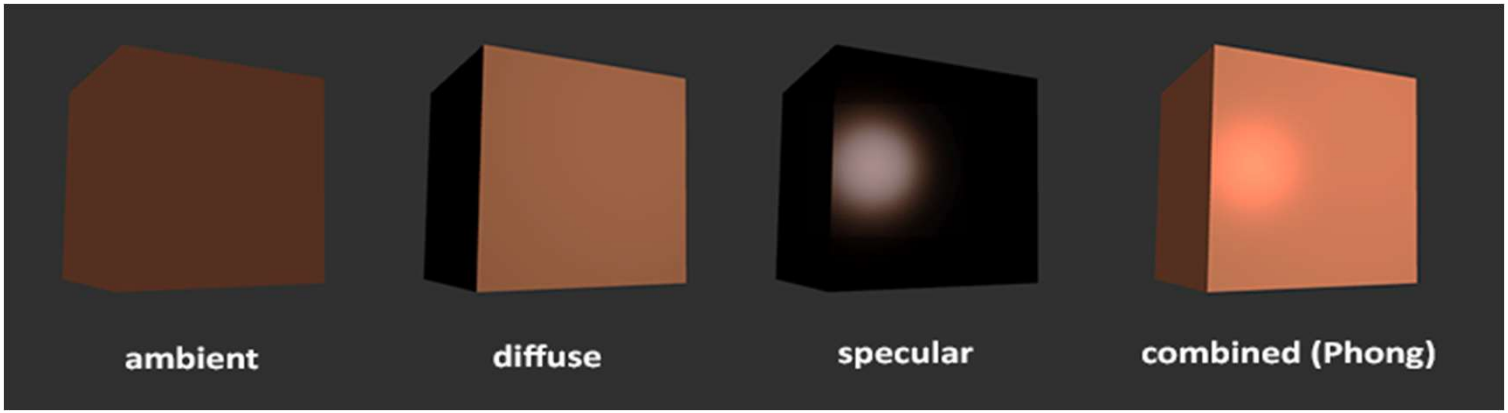
- Προσθήκη βοηθητικών συναρτήσεων στην κλάση Shader.h

https://learnopengl.com/code_viewer_gh.php?code=includes/learnopengl/shader.h

Μοντέλο Phong

- **Μοντέλα φωτισμού**: προσέγγιση της πραγματικότητας με απλά μαθηματικά μοντέλα

- **Μοντέλο Phong**: έμμεσος (ambient), διάχυτος (diffuse) και κατευθυνόμενος (specular) φωτισμός



Μοντέλο Phong

- **Έμμεσος φωτισμός**: προέρχεται από ανακλάσεις του φωτός στο περιβάλλον

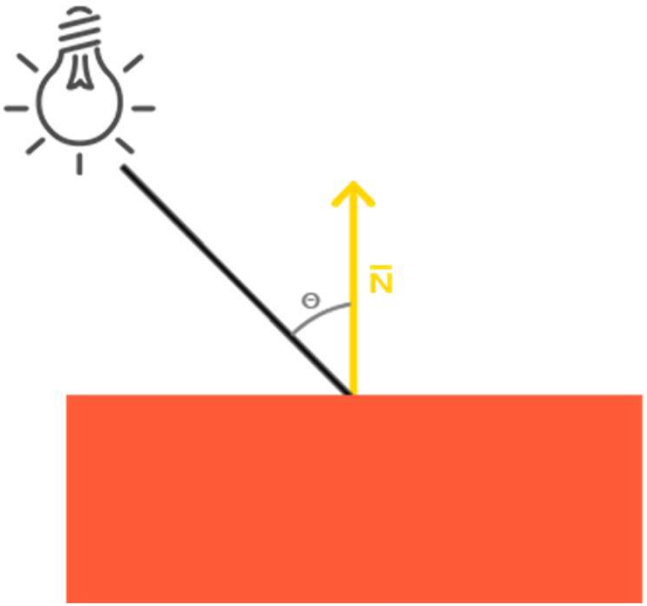
```
float ambientStrength = 0.1;
 vec3 ambient = ambientStrength * lightColor;

vec3 result = ambient * objectColor;
gl_FragColor = vec4(result, 1.0);
```

Μοντέλο Phong

- **Διάχυτος φωτισμός**: προέρχεται από την ευθυγράμμιση της επιφάνειας με τη φωτεινή πηγή

Για να τον υπολογίσουμε χρειάζονται:

- Η κατεύθυνση της ακτίνας του φωτός
- Η κατεύθυνση (το κανονικό διάνυσμα) της επιφάνειας



Κύβος με κανονικά διανύσματα

```
float vertices[] = {
    -0.5f, -0.5f, -0.5f,  0.0f,  0.0f, -1.0f,
     0.5f, -0.5f, -0.5f,  0.0f,  0.0f, -1.0f,
     0.5f,  0.5f, -0.5f,  0.0f,  0.0f, -1.0f,
     0.5f,  0.5f, -0.5f,  0.0f,  0.0f, -1.0f,
    -0.5f,  0.5f, -0.5f,  0.0f,  0.0f, -1.0f,
    -0.5f, -0.5f, -0.5f,  0.0f,  0.0f, -1.0f,

    -0.5f, -0.5f,  0.5f,  0.0f,  0.0f, 1.0f,
     0.5f, -0.5f,  0.5f,  0.0f,  0.0f, 1.0f,
     0.5f,  0.5f,  0.5f,  0.0f,  0.0f, 1.0f,
     0.5f,  0.5f,  0.5f,  0.0f,  0.0f, 1.0f,
    -0.5f,  0.5f,  0.5f,  0.0f,  0.0f, 1.0f,
    -0.5f, -0.5f,  0.5f,  0.0f,  0.0f, 1.0f,

    -0.5f,  0.5f,  0.5f, -1.0f,  0.0f,  0.0f,
    -0.5f,  0.5f, -0.5f, -1.0f,  0.0f,  0.0f,
    -0.5f, -0.5f, -0.5f, -1.0f,  0.0f,  0.0f,
    -0.5f, -0.5f, -0.5f, -1.0f,  0.0f,  0.0f,
    -0.5f, -0.5f,  0.5f, -1.0f,  0.0f,  0.0f,
    -0.5f,  0.5f,  0.5f, -1.0f,  0.0f,  0.0f,

     0.5f,  0.5f,  0.5f,  1.0f,  0.0f,  0.0f,
     0.5f,  0.5f, -0.5f,  1.0f,  0.0f,  0.0f,
     0.5f, -0.5f, -0.5f,  1.0f,  0.0f,  0.0f,
     0.5f, -0.5f, -0.5f,  1.0f,  0.0f,  0.0f,
     0.5f, -0.5f,  0.5f,  1.0f,  0.0f,  0.0f,
     0.5f,  0.5f,  0.5f,  1.0f,  0.0f,  0.0f,

    -0.5f, -0.5f, -0.5f,  0.0f, -1.0f,  0.0f,
     0.5f, -0.5f, -0.5f,  0.0f, -1.0f,  0.0f,
     0.5f, -0.5f,  0.5f,  0.0f, -1.0f,  0.0f,
     0.5f, -0.5f,  0.5f,  0.0f, -1.0f,  0.0f,
    -0.5f, -0.5f,  0.5f,  0.0f, -1.0f,  0.0f,
    -0.5f, -0.5f, -0.5f,  0.0f, -1.0f,  0.0f,

    -0.5f,  0.5f, -0.5f,  0.0f,  1.0f,  0.0f,
     0.5f,  0.5f, -0.5f,  0.0f,  1.0f,  0.0f,
     0.5f,  0.5f,  0.5f,  0.0f,  1.0f,  0.0f,
     0.5f,  0.5f,  0.5f,  0.0f,  1.0f,  0.0f,
    -0.5f,  0.5f,  0.5f,  0.0f,  1.0f,  0.0f,
    -0.5f,  0.5f, -0.5f,  0.0f,  1.0f,  0.0f
};
```

Μοντέλο Phong

- Η κατεύθυνση της ακτίνας του φωτός:

```
vec3 lightDir = normalize(lightPos - FragPos);
```

- lightPos → η θέση της φωτεινής πηγής:

```
uniform vec3 lightPos;
```

- FragPos → η θέση του αντικειμένου στην οποία υπολογίζουμε το χρώμα:

```
FragPos = vec3(model * vec4(in_position, 1.0));
```

Μοντέλο Phong

- Προσθήκη στον Vertex shader:

```
layout (location = 1) in vec3 in_normal;
uniform mat4 model;
out vec3 FragPos;
out vec3 Normal;
```

- Στη συνάρτηση main():

```
gl_Position = transform* vec4(in_position, 1.0);
FragPos = vec3(model * vec4(in_position, 1.0));
Normal = in_normal;
```

Μοντέλο Phong

- Προσθήκη στον Fragment shader:

```
uniform vec3 lightPos;
in vec3 FragPos;
in vec3 Normal;
```

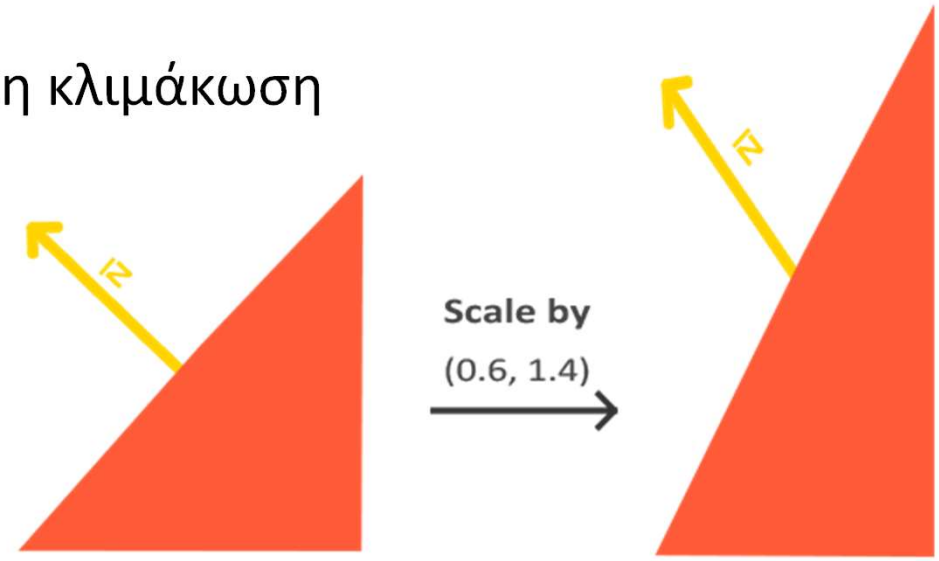
- Στη συνάρτηση main():

```
vec3 norm = normalize(Normal);
vec3 lightDir = normalize(lightPos - FragPos);

float diff = max(dot(norm, lightDir), 0.0);
vec3 diffuse = diff * lightColor;

vec3 result = (ambient + diffuse) * objectColor;
gl_FragColor = vec4(result, 1.0);
```

Μοντέλο Phong

- Οι υπολογισμοί γίνονται στο σ.σ. του κόσμου
- Μετασχηματισμός των κανονικών διανυσμάτων??

- Παράδειγμα: μη – ομοιόμορφη κλιμάκωση



Μοντέλο Phong

- Τα κανονικά διανύσματα είναι διανύσματα κατεύθυνσης, άρα **w = 0**

- **w = 0** → δεν αλλάζουν όταν το αντικείμενο μετατοπίζεται, μόνο με τους μετασχηματισμούς της περιστροφής και κλιμάκωσης

- **Πίνακας normal**: ειδικός για τους μετασχηματισμούς των κανονικών διανυσμάτων, είναι ο ανάστροφος του αντίστροφου του πάνω-δεξιά 3x3 πίνακα model

Μοντέλο Phong

- Υπολογισμός του πίνακα normal:

```
glm::mat3 normal = glm::transpose(glm::inverse(glm::mat3(model)));
```

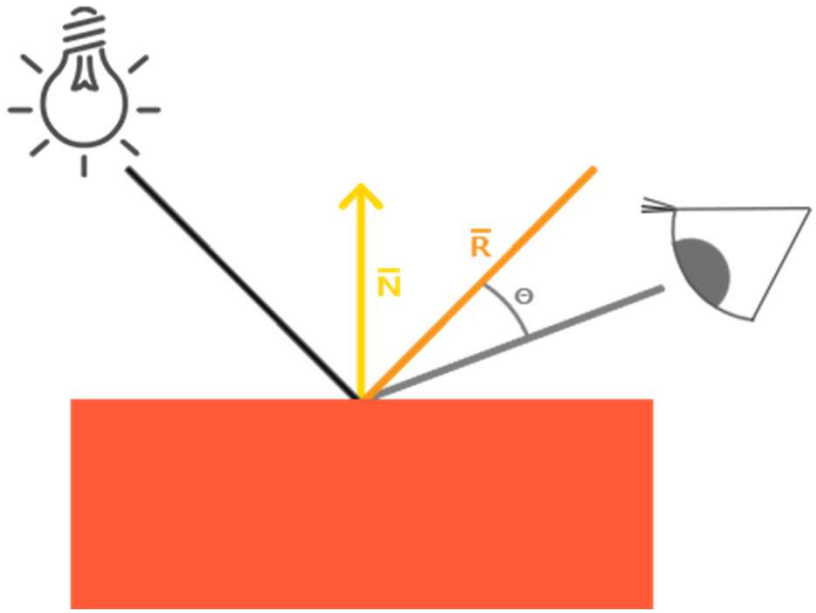
- Προσθήκη στον Vertex shader:

```
uniform mat3 normal;

Normal = normal * in_normal;
```

Μοντέλο Phong

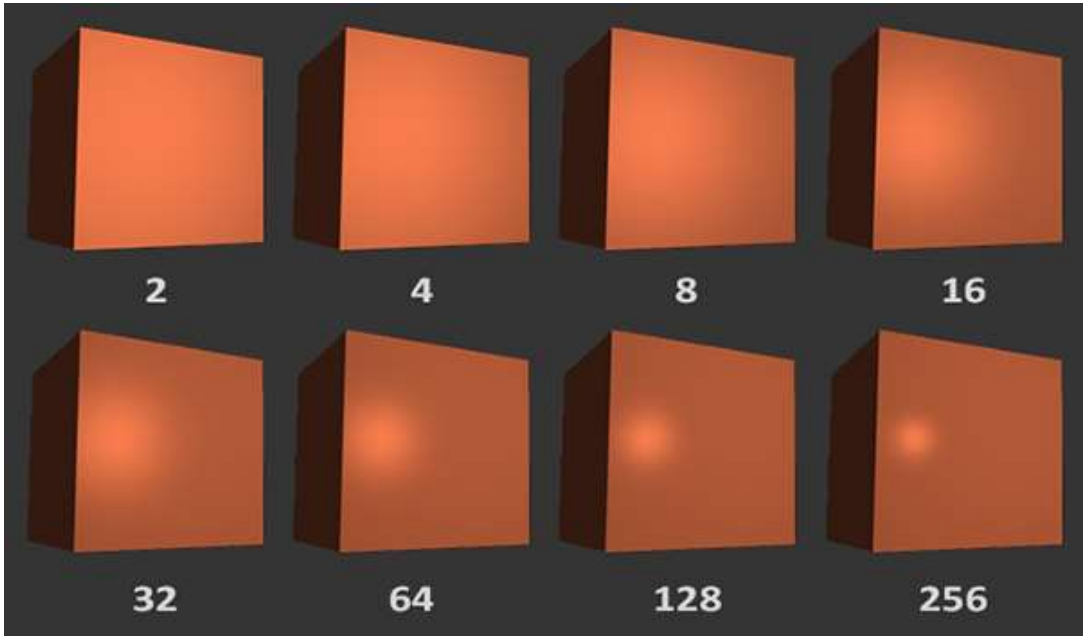
- **Κατευθυνόμενος φωτισμός**: προέρχεται από την ευθυγράμμιση του θεατή με την ανάκλαση της φωτεινής πηγής στην επιφάνεια

- Το αντικείμενο λειτουργεί σαν καθρέφτης



Μοντέλο Phong

- Όσο πιο γυαλιστερό είναι το αντικείμενο (shininess), τόσο πιο συγκεντρωμένη θα είναι η κατευθυνόμενη ανάκλαση του φωτός



Μοντέλο Phong

- Προσθήκη στον Fragment shader:

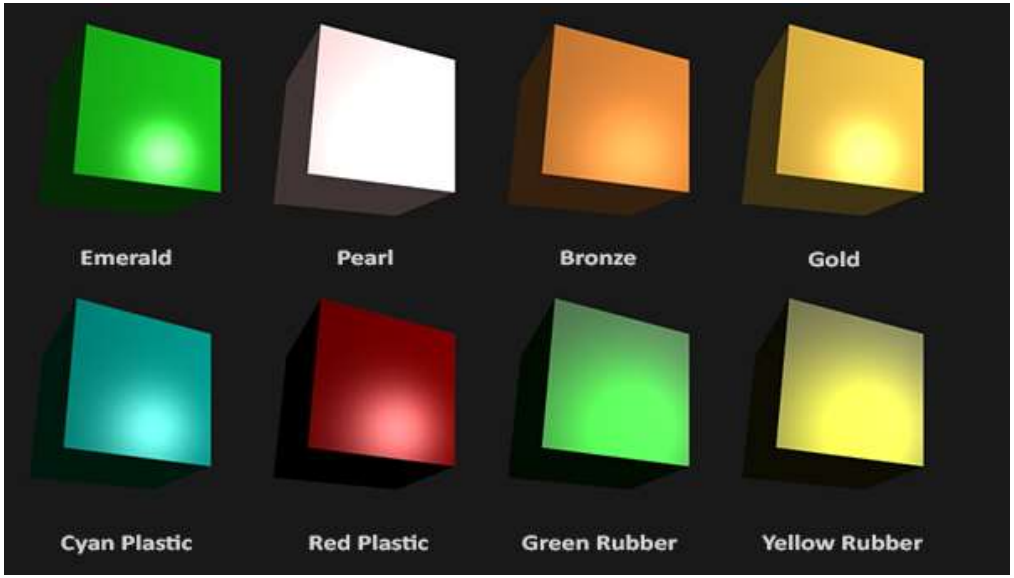
```
uniform vec3 viewPos;
```

- Στη συνάρτηση main():

```
vec3 viewDir = normalize(viewPos - FragPos);
vec3 reflectDir = reflect(-lightDir, norm);

float spec = pow(max(dot(viewDir, reflectDir), 0.0), 32.0); //shininess = 32.0
float specularStrength = 0.5;
vec3 specular = specularStrength * spec * lightColor;

vec3 result = (ambient + diffuse + specular) * objectColor;
gl_FragColor = vec4(result, 1.0);
```

Υλικά

- Διαφορετικά υλικά → διαφορετικές εντάσεις έμμεσης, διάχυτης και κατευθυνόμενης συνιστώσας του χρώματος



- http://devernay.free.fr/cours/opengl/materials.html

Υλικά

- Στον Fragment shader:

```
struct Material {
    vec3 ambient;
    vec3 diffuse;
    vec3 specular;
    float shininess;
};

uniform Material material;
```

Υλικά

- Στον Fragment shader:

```
// ambient
vec3 ambient = lightColor * material.ambient;

// diffuse
vec3 diffuse = lightColor * (diff * material.diffuse);

//specular
float spec = pow(max(dot(viewDir, reflectDir), 0.0), material.shininess);
vec3 specular = lightColor * (spec * material.specular);
```

Υλικά

- Για να χρησιμοποιήσουμε υλικά από τον πίνακα του link:
shininess * 128.0

- Ένας κύβος από γαλάζιο πλαστικό:

```
objShader.setVec3("material.ambient", glm::vec3(0.0f, 0.1f, 0.06f));
objShader.setVec3("material.diffuse", glm::vec3(0.0f, 0.51f, 0.51f));
objShader.setVec3("material.specular", glm::vec3(0.51f, 0.51f, 0.51f));
objShader.setFloat("material.shininess", 0.25);
```

Υλικά

- Η ένταση του φωτός είναι μεγάλη, μπορούμε να την ελέγχουμε:

```
struct Light {
    vec3 position;

    vec3 ambient;
    vec3 diffuse;
    vec3 specular;
};
uniform Light light;
```

Υλικά

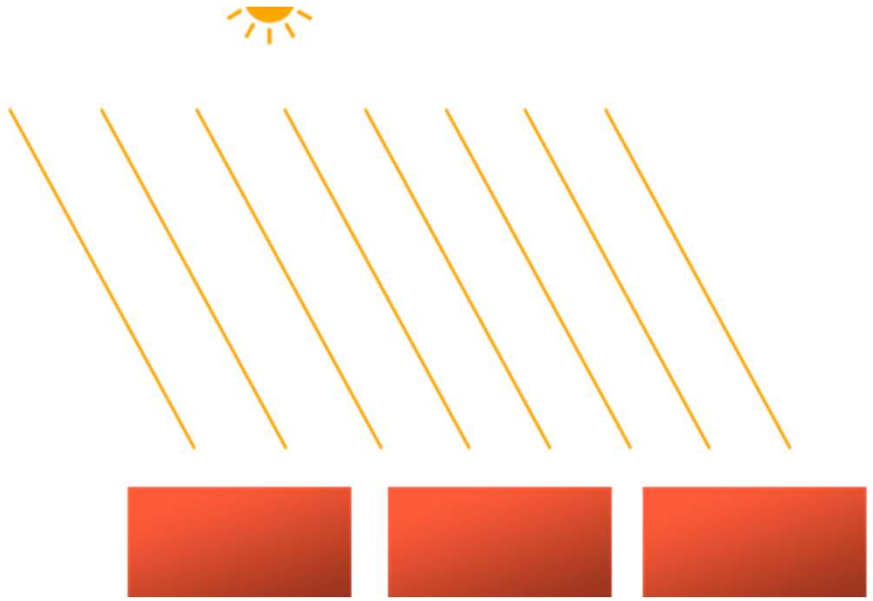
- Αλλαγή στον Fragment shader:

```
vec3 ambient  = light.ambient * material.ambient;
vec3 diffuse  = light.diffuse * (diff * material.diffuse);
vec3 specular = light.specular * (spec * material.specular);
```

- Οι συνιστώσες material.ambient και material.diffuse έχουν το ίδιο χρώμα, η ένταση καθορίζεται από την ένταση του φωτός

Είδη φωτεινών πηγών

- **Κατευθυντική**: η κατεύθυνση του φωτός είναι ίδια για όλα τα σημεία στο χώρο



Είδη φωτεινών πηγών

- Στον Fragment shader:

```
struct Light {
    // vec3 position; // Not needed when using directional lights.
    vec3 direction;

    vec3 ambient;
    vec3 diffuse;
    vec3 specular;
};
```

- Στη συνάρτηση main():

```
vec3 lightDir = normalize(-light.direction);
```

Είδη φωτεινών πηγών

- Η κατεύθυνση του φωτός:

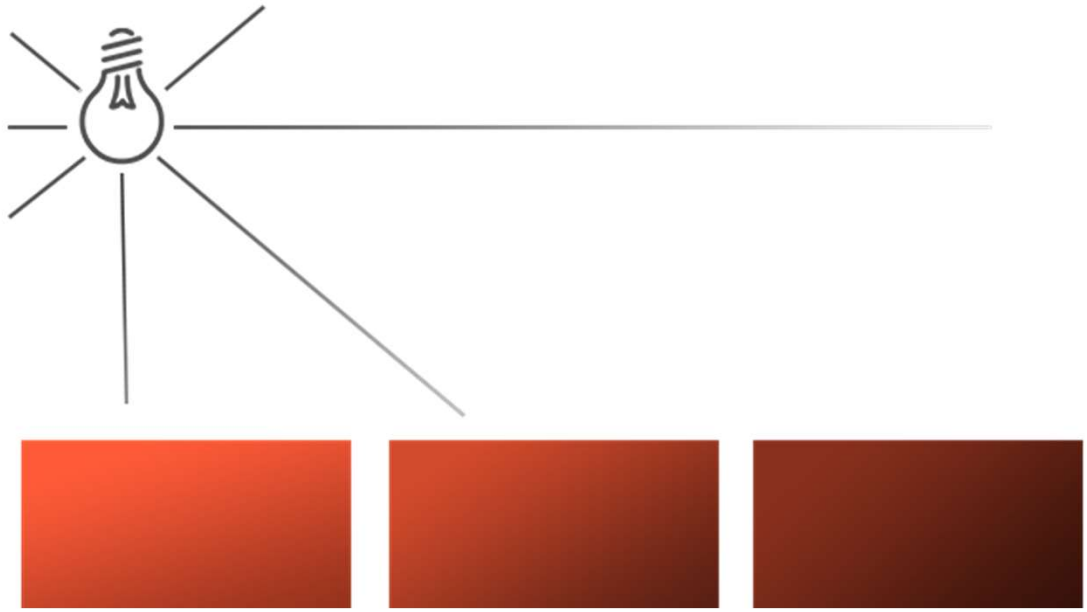
```
lightingShader.setVec3("light.direction", -0.2f, -1.0f, -0.3f);
```

- Επειδή είναι διάνυσμα, **w = 0**

- Μπορούμε να καταλάβουμε αν η πηγή είναι σημειακή ή κατευθυντική αν **w = 1** ή **w = 0**

Είδη φωτεινών πηγών

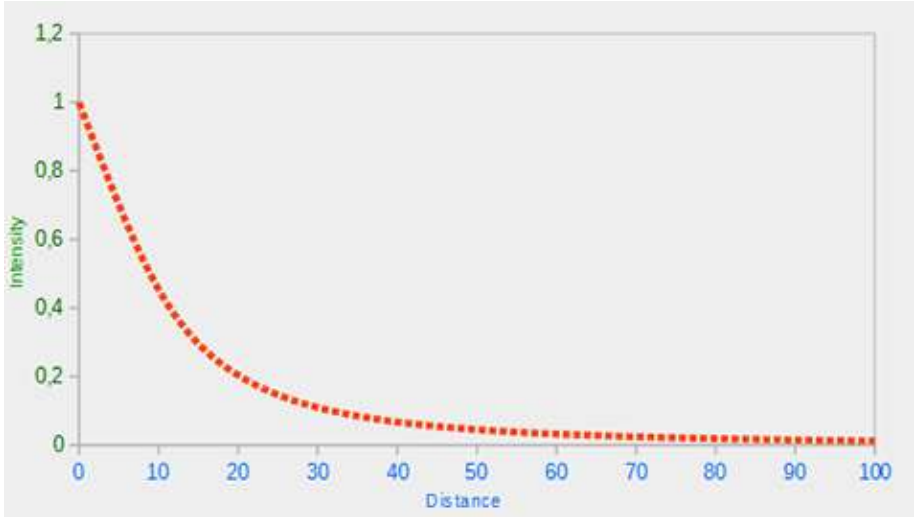
- **Σημειακή**: οι ακτίνες του φωτός ξεκινούν από ένα σημείο και απλώνονται στο χώρο



Είδη φωτεινών πηγών

- Θέλουμε η ένταση του φωτός να μειώνεται με την απόσταση από την φωτεινή πηγή

- Σταθερές εξασθένησης (attenuation): $$F_{att} = \frac{1.0}{K_c + K_l * d + K_q * d^2}$$



Είδη φωτεινών πηγών

- Ανάλογα με την ισχύ της φωτεινής πηγής, το φως θα καλύψει μικρότερη ή μεγαλύτερη απόσταση

- Ανάλογα με την απόσταση διαλέγουμε τις σταθερές εξασθένησης

Distance	Constant	Linear	Quadratic
7	1.0	0.7	1.8
13	1.0	0.35	0.44
20	1.0	0.22	0.20
32	1.0	0.14	0.07
50	1.0	0.09	0.032
65	1.0	0.07	0.017
100	1.0	0.045	0.0075
160	1.0	0.027	0.0028
200	1.0	0.022	0.0019
325	1.0	0.014	0.0007
600	1.0	0.007	0.0002
3250	1.0	0.0014	0.000007

Είδη φωτεινών πηγών

- Στον Fragment shader:

```
struct Light {
    vec3 position;

    vec3 ambient;
    vec3 diffuse;
    vec3 specular;

    float constant;
    float linear;
    float quadratic;
};
```

Είδη φωτεινών πηγών

- Στη συνάρτηση main():

```
float distance    = length(light.position - FragPos);
float attenuation = 1.0 / (light.constant + light.linear * distance +
                           light.quadratic * (distance * distance));


ambient  *= attenuation;
diffuse  *= attenuation;
specular *= attenuation;
//if only one light source, ambient can be kept constant
```