Лекция 6. Материалы и источники света

Илья Макаров

ИТМО

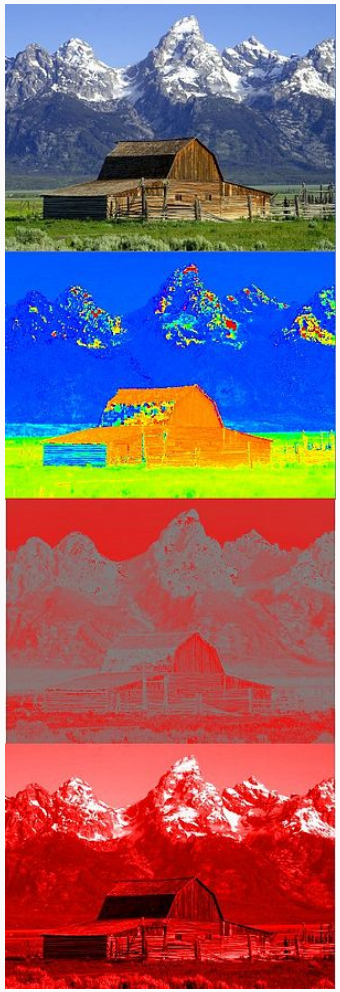
19 октября 2022

Санкт-Петербург

Представление цветов

- **RGB**
- **HSV** - тон, насыщенность и яркость
- **HSL** - тон, насыщенность и светлота
- и другие...

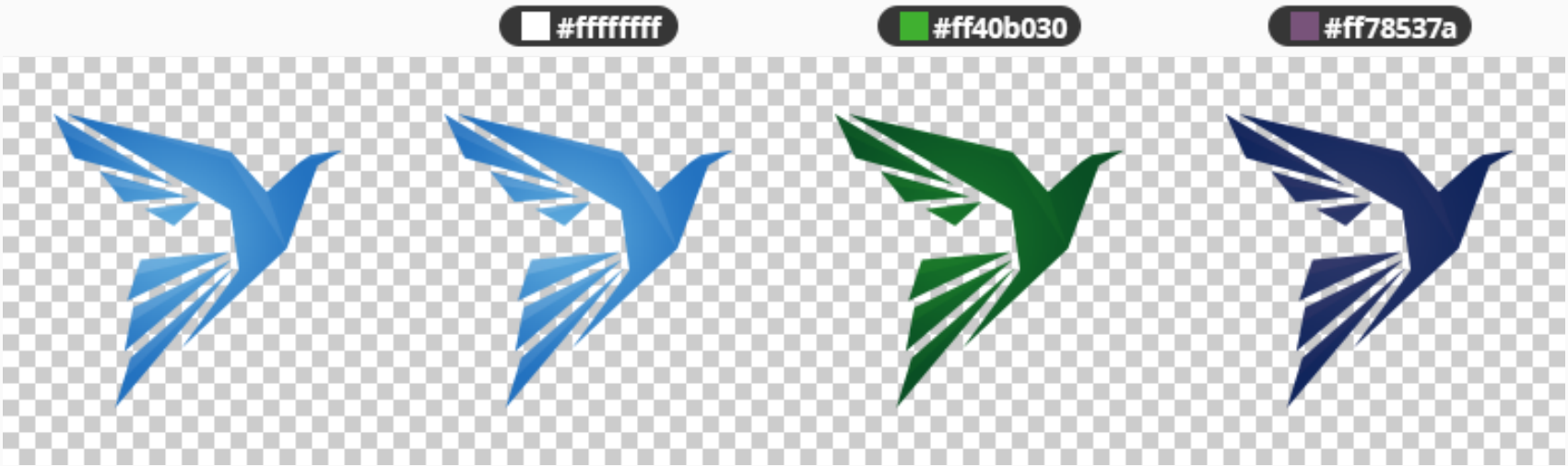
Представление цветов HSL



Смешивание цветов

multiply

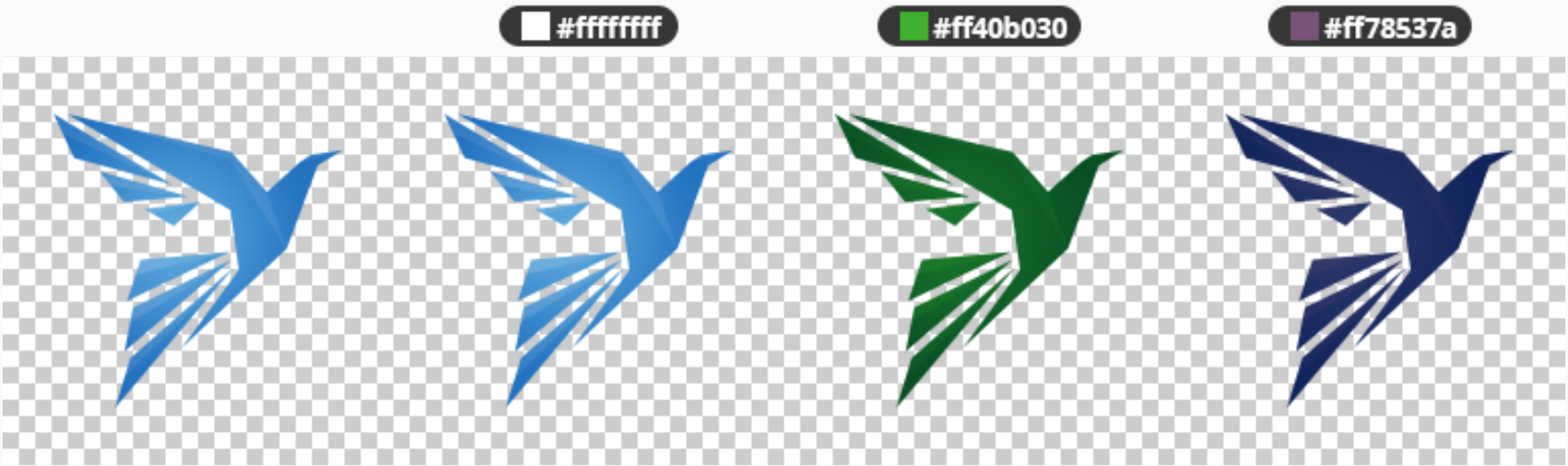
- смешивает цвета, перемножая их. Получаемый цвет всегда будет темнее базового.



Смешивание цветов

multiply

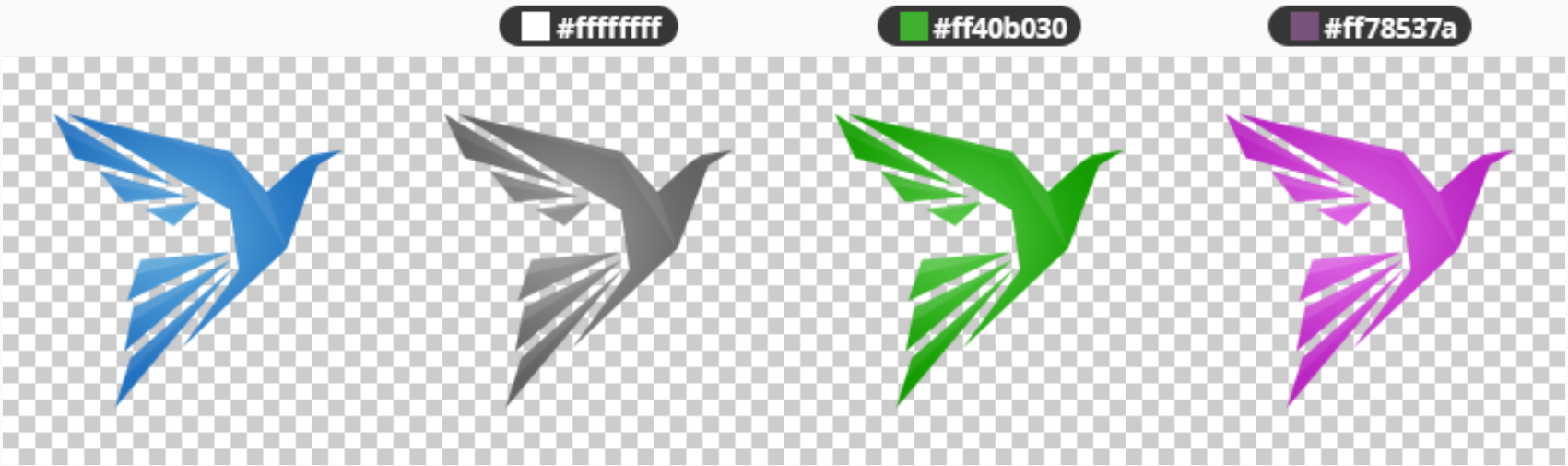
- смешивает цвета, перемножая их. Получаемый цвет всегда будет темнее базового.



Смешивание цветов

hue

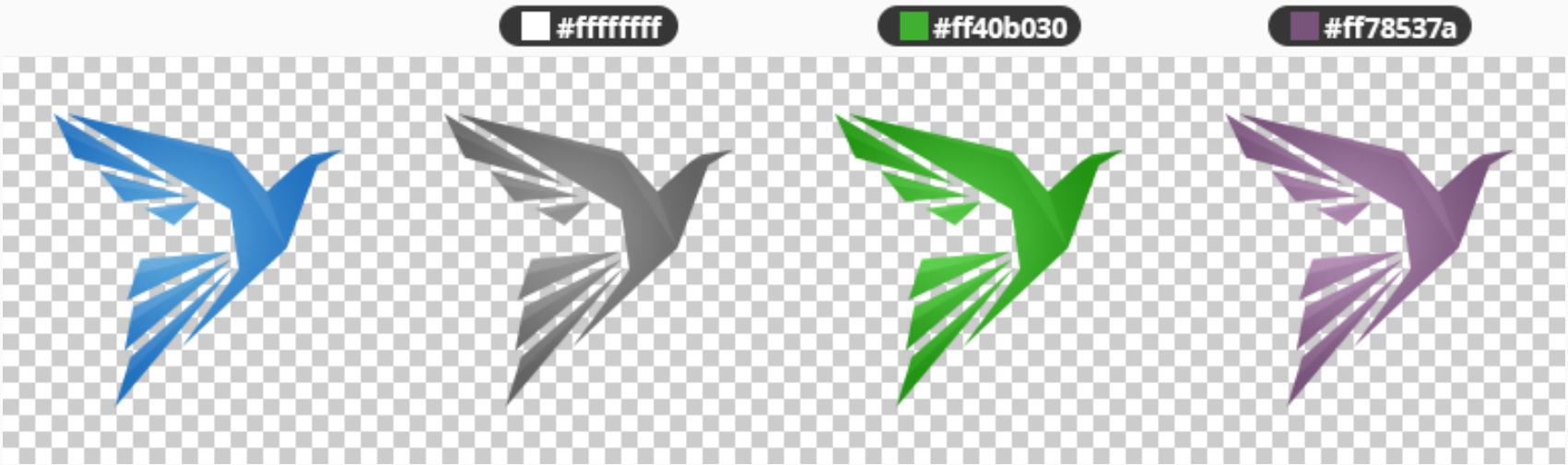
- смешивает цвета, изменяя оттенок базового цвета на оттенок смешиваемого, сохраняя насыщенность и светлоту базового цвета.



Смешивание цветов

color

- смешивает цвета, изменяя оттенок и насыщенность базового цвета на оттенок и насыщенность смешиваемого, сохраняя светлоту базового цвета.



Смешивание цветов

difference

- смешивает цвета, вычитая меньшие значения из больших. Смешение с белым цветом инвертирует цвет, с черным не дает изменений.

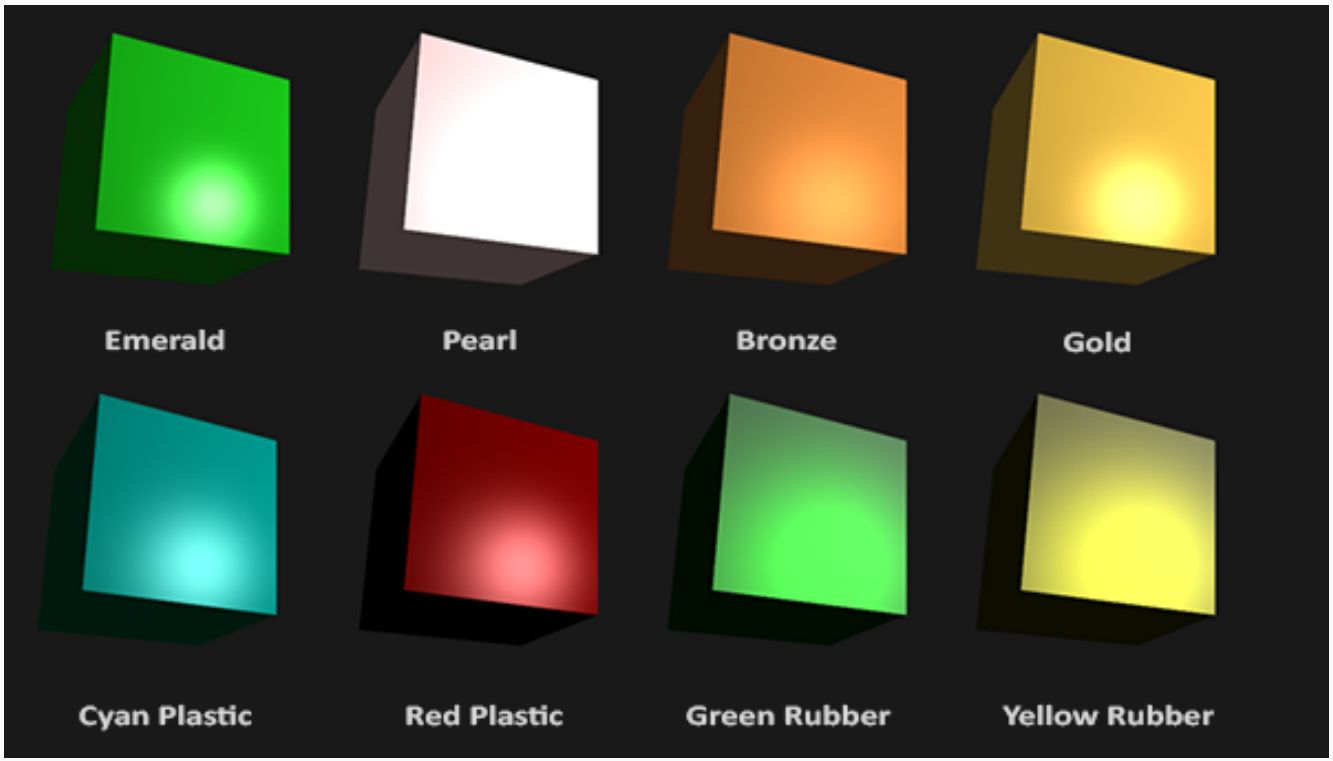
Смешивание цветов

fill

- смешивает цвета, заливая непрозрачные пиксели базового цвета смешиваемым. Может использоваться для заливки прозрачных текстур с абстрактными формами.

Материалы

Каждый объект по разному реагирует на свет.



Материалы

Удобно параметризовать конкретные объекты следующим образом.

```
#version 330 core

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

uniform Material material;
```

Компоненты материала

- Вектор **ambient** определяет, какой цвет объект отражает под фоновым освещением. Обычно это цвет самого объекта.
- Вектор **diffuse** определяет цвет объекта под рассеянным освещением. Также, как и фоновый, он определяет желаемый цвет объекта.
- Вектор **specular** устанавливает цвет блика на объекте, а переменная **shininess** — радиус этого блика.

Материалы

В коде шейдера необходимо учитывать материал:

```
void main() {
    // ambient
    vec3 ambient = lightColor * material.ambient;

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

    // specular
    // ...
    vec3 specular = lightColor
        * (spec * material.specular);
}
```

Свойства источника света

Структура подобная материалу:

```
#version 330 core

struct Light {
    vec3 position;

    vec3 ambient;
    vec3 diffuse;
    vec3 specular;
};

uniform Light light;
```

Свойства источника света

Параметризуем шейдер:

```
vec3 ambient  = light.ambient * material.ambient;

vec3 diffuse  = light.diffuse
    * (diff * material.diffuse);

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

Текстурные карты

Только цвета не достаточно для представления более сложных объектов. Для представления таких объектов как правило используют текстурные карты.

- **diffuse** - диффузная компонента, задает цвет;
- **specular** - карта бликов, отражает силу бликов;
- **normal** - карта нормалей (детали самой поверхности);
- И другие.

Текстурные карты

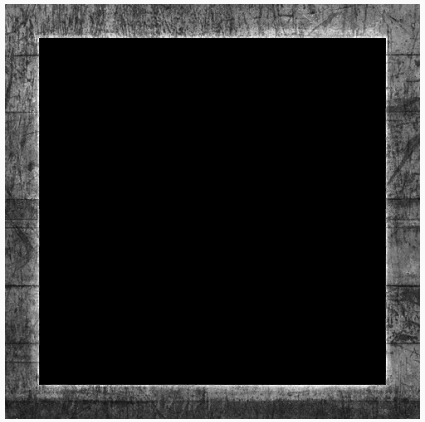
В коде шейдера:

```
#version 330 core

struct Material {
    sampler2D diffuse;
    sampler2D specular;
    sampler2D normal;
    float shininess;
};

uniform Material material;
```

Текстурные карты

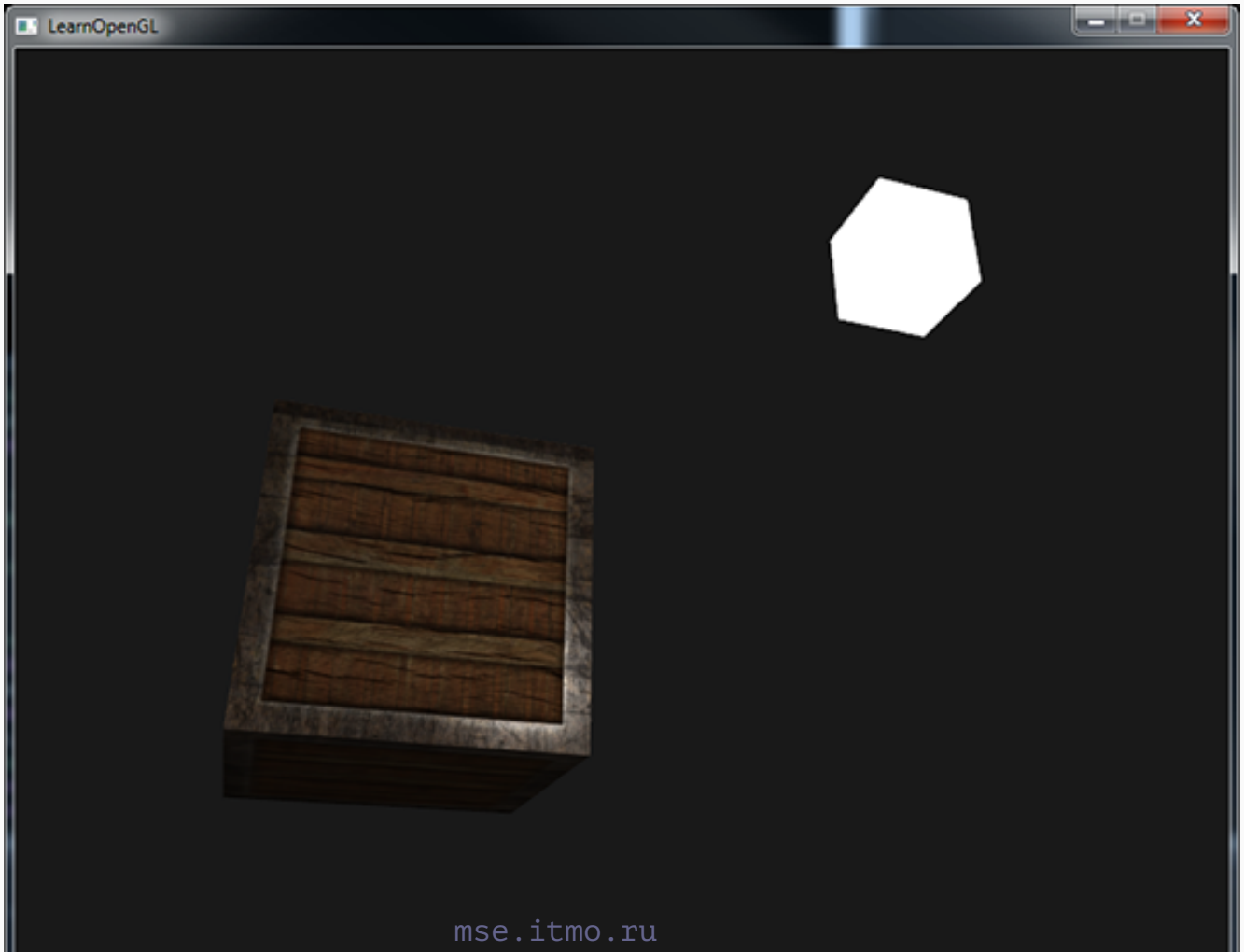
Диффузная карта

Карта бликов



Текстурные карты

Их комбинация дает следующий результат

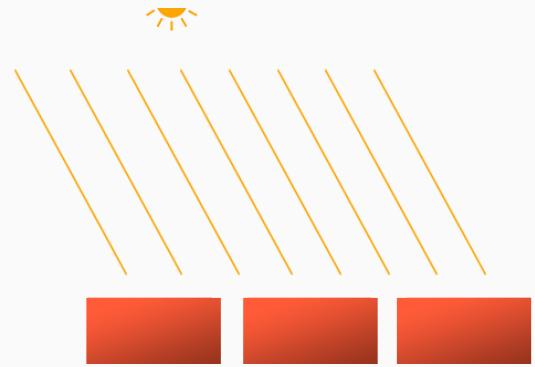


Источники света

Основные типы источников:

- **направленный** (directional)
- **точечный** (point)
- **прожектор** (spot)

Направленный источник

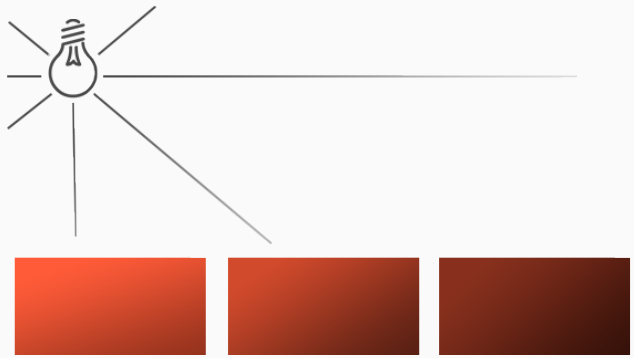


Параметризуется:

```
struct Light {
    // no longer necessary when using directional lights.
    // vec3 position;
    vec3 direction;

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

Точечный источник

Светит во все стороны, однако интенсивность затухает.

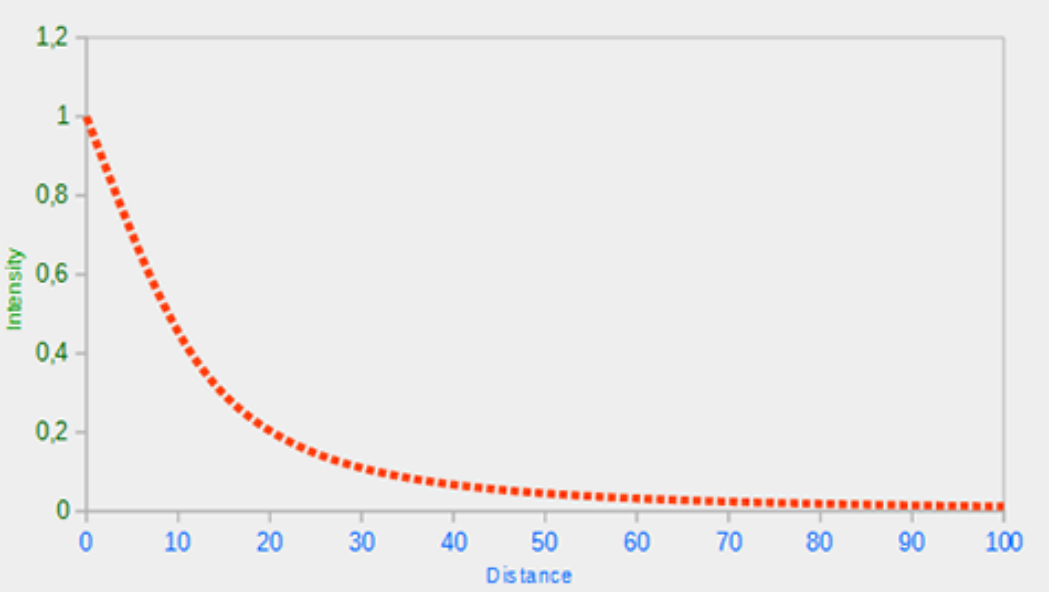


Как выбрать функцию для интенсивности?

- линейно
- квадратично
- иначе

Точечный источник

Лучше всего себя показывает следующая зависимость интенсивности от расстояния $I(d) = \frac{1}{K_c + K_l \cdot d + K_q \cdot d^2}$



Точечный источник

В шейдере же параметризуется:

```
struct Light {
    vec3 position;

    vec3 ambient;
    vec3 diffuse;
    vec3 specular;

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

Точечный источник

Для расчета затухания:

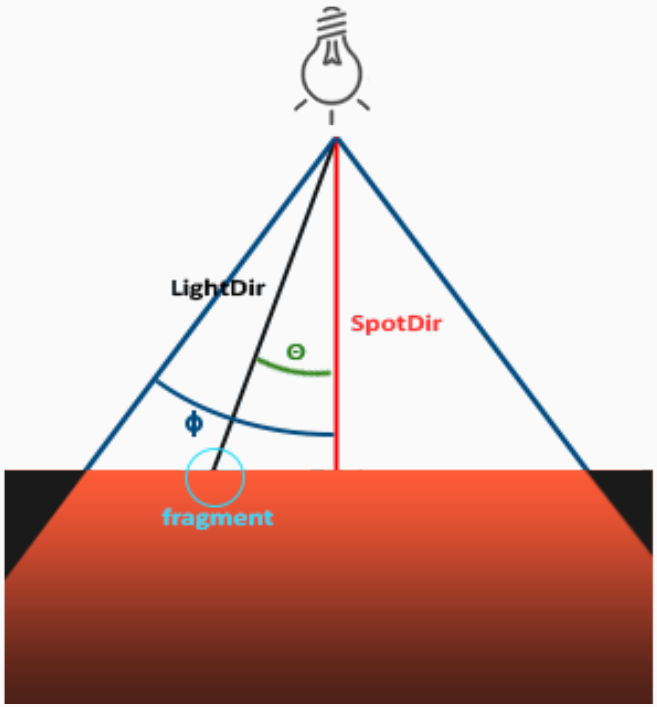
```
float distance    = length(light.position - FragPos);
float attenuation = light.constant
    + light.linear * distance
    + light.quadratic * (distance * distance);
```

Применим к компонентам:

```
ambient  *= 1. / attenuation;
diffuse  *= 1. / attenuation;
specular *= 1. / attenuation;
```

Прожектор

Направленный пучок света. Ограниченный конусом



Прожектор

В шейдере параметризуется:

```
struct Light {
    vec3  position;
    vec3  direction;
    float cutOff;
    float outerCutOff;
    ...
};
```

Прожектор

Для расчета света используем следующее соотношение:

```
float theta = dot(lightDir, normalize(-light.direction));

if(theta > light.cutOff) {
    // do lighting calculations
}
else {
    // else, use ambient light so scene
    // isn't completely dark outside the spotlight.
}
```

Прожектор

Чтобы смягчить переход, необходимо задать затухание между внутренним и внешним конусами $I(\theta) = \frac{\theta - \gamma}{\epsilon}, \epsilon = \phi - \gamma$

- γ - внешний угол
- ϕ - внутренний
- ϵ - разница углов

```
float theta = dot(lightDir,
    normalize(-light.direction));
float epsilon = light.cutOff - light.outerCutOff;
float intensity = clamp(
    (theta - light.outerCutOff) / epsilon, 0.0, 1.0);

...

diffuse  *= intensity;
specular *= intensity;
```