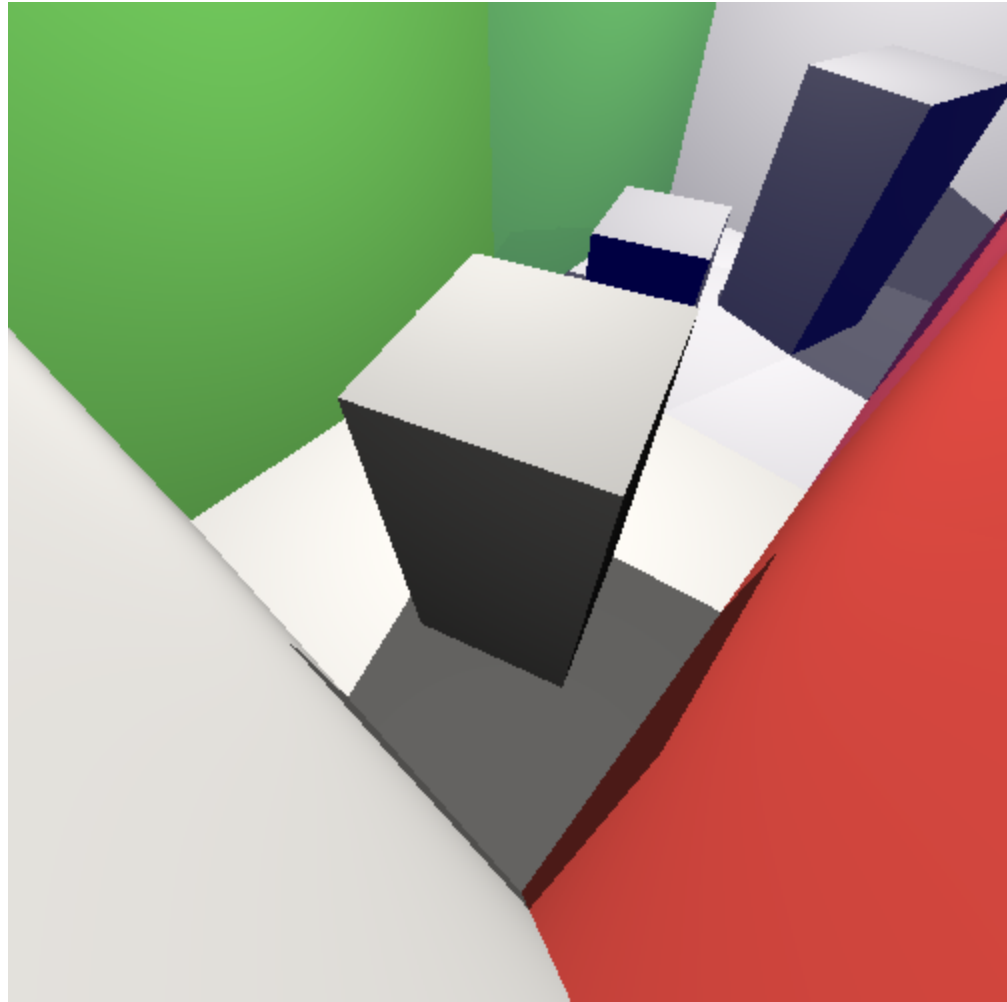
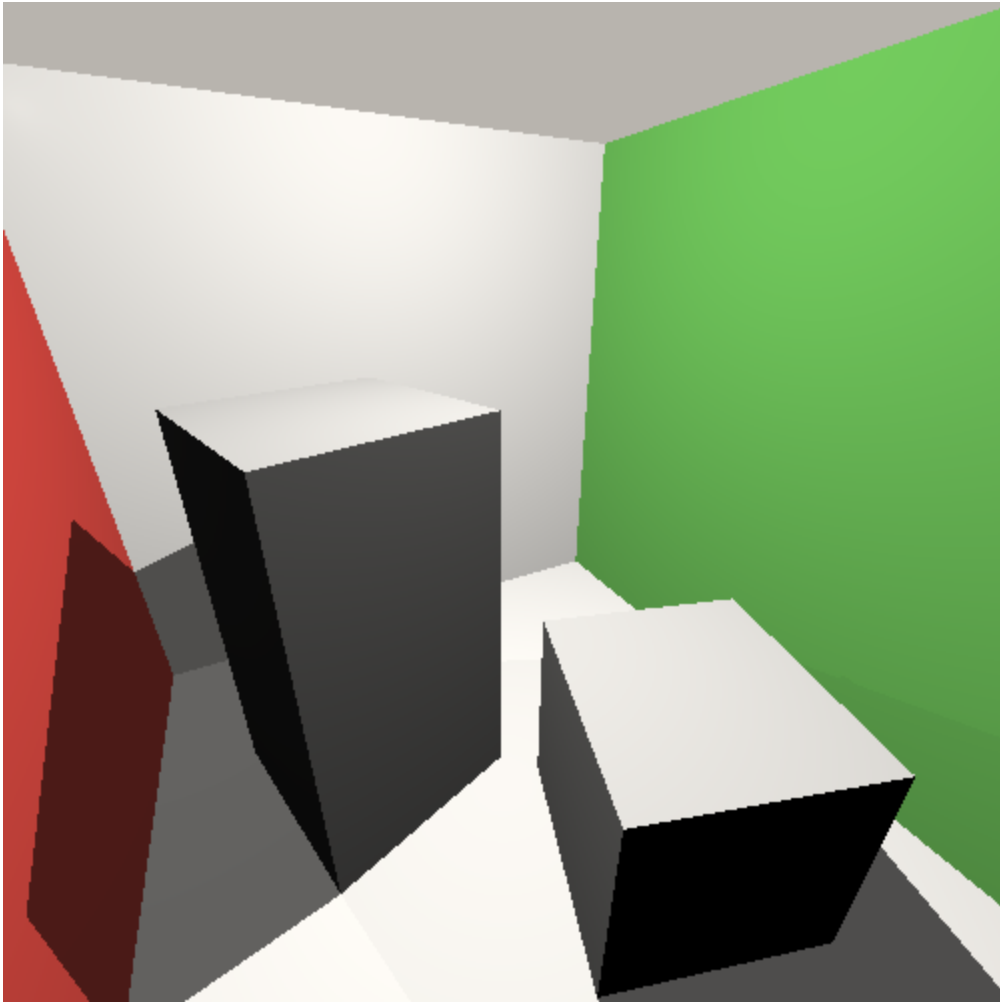
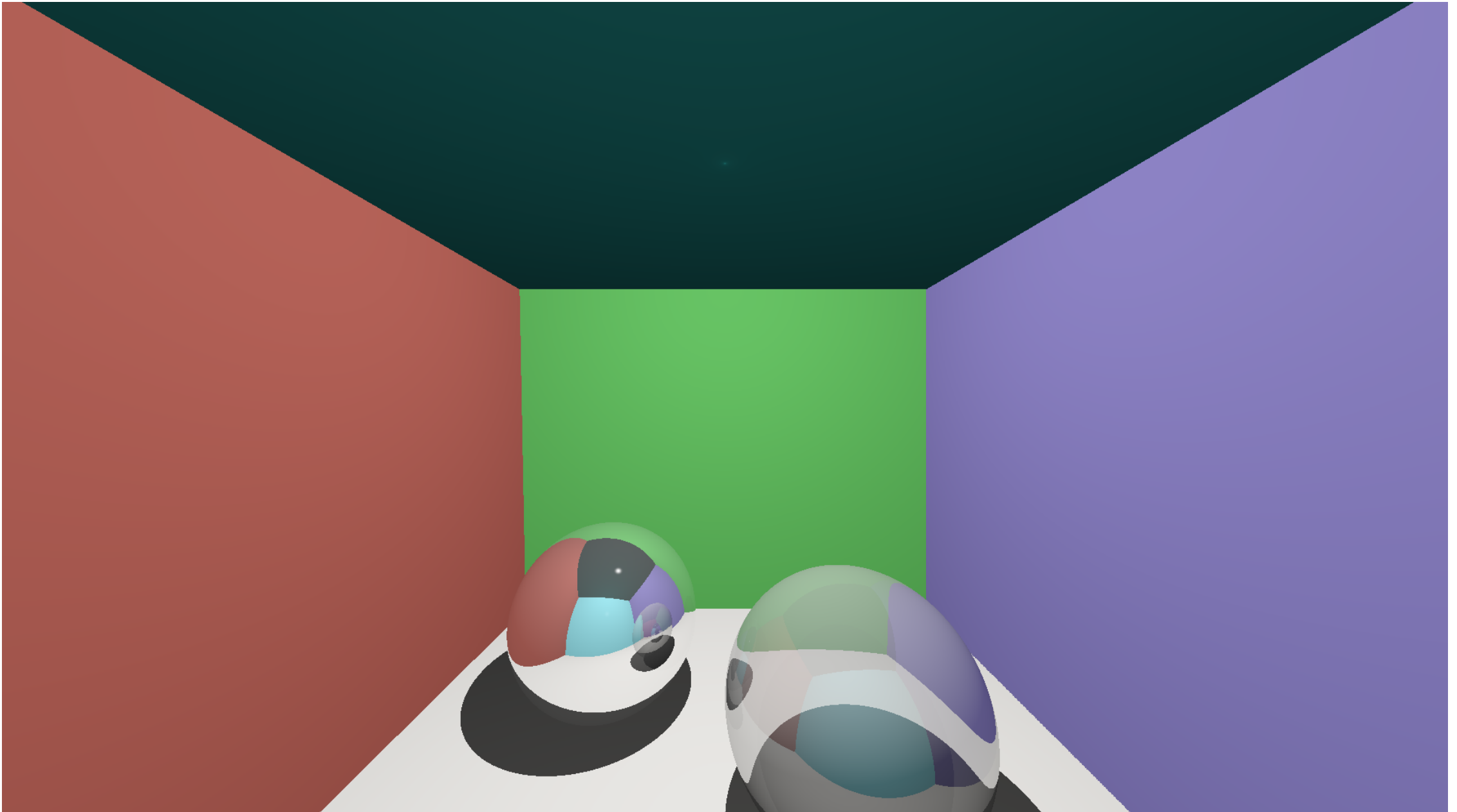


Homework 1. Raytracer

Простейшие сцены



Отражения и преломления



Бонус 1



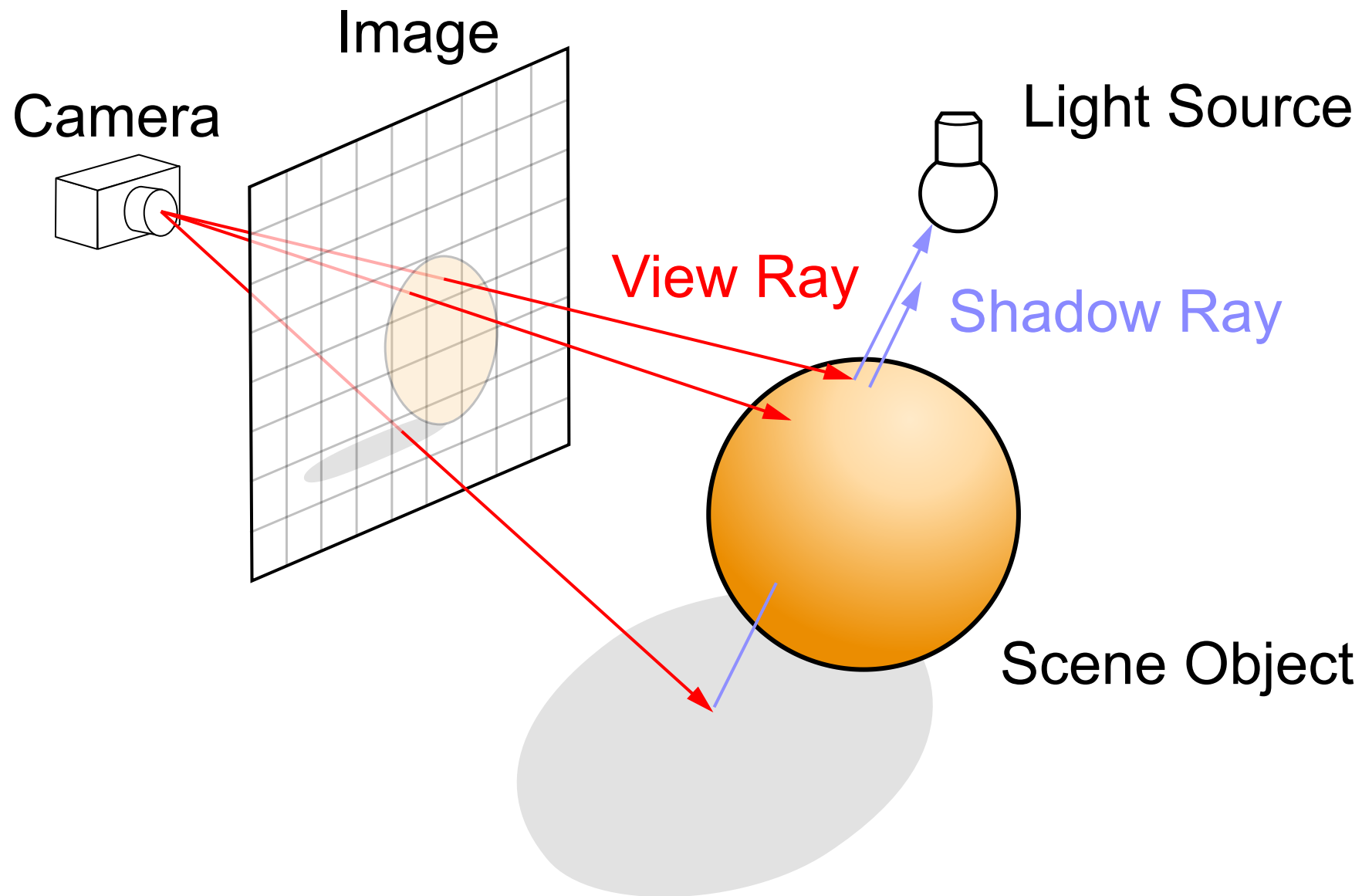
Бонус 2



Базовый алгоритм

- Ставим камеру и экран напротив
- Проводим луч из позиции камеры через "пиксель" (x, y) экрана
- Находим точку пересечения p этого луча со сценой (ближайший объект)
- Считаем освещенность в точке p , записываем ее как значения пикселя в (x, y)

Графически



Освещенность в точке

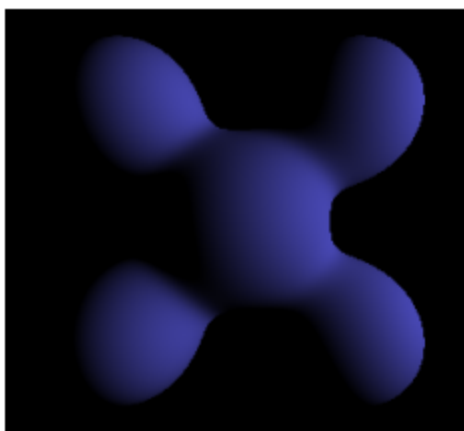
- В нашем случае будут только простейшие точечные источники света
- Проитерируем по всем источникам света и просуммируем вклад каждого в освещенность в точке p
- Как узнать дает ли источник света L вклад в освещенность в p ? Проведем луч из p в L , если на пути есть какой-то объект, то p находится в тени относительно L .
- Как посчитать сам этот вклад?

Модель Фонга



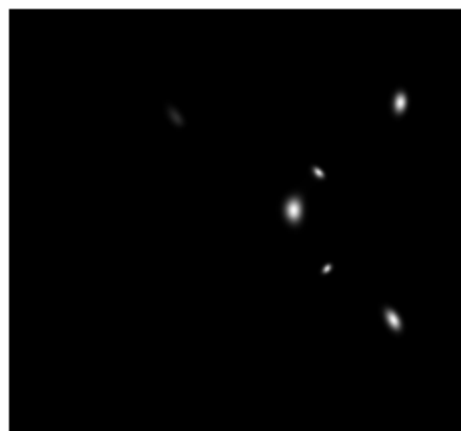
Ambient

+



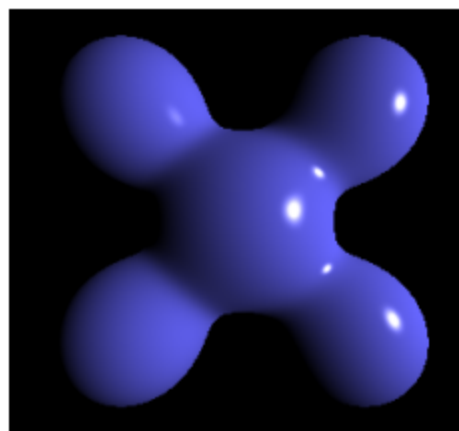
Diffuse

+



Specular

=



Phong Reflection

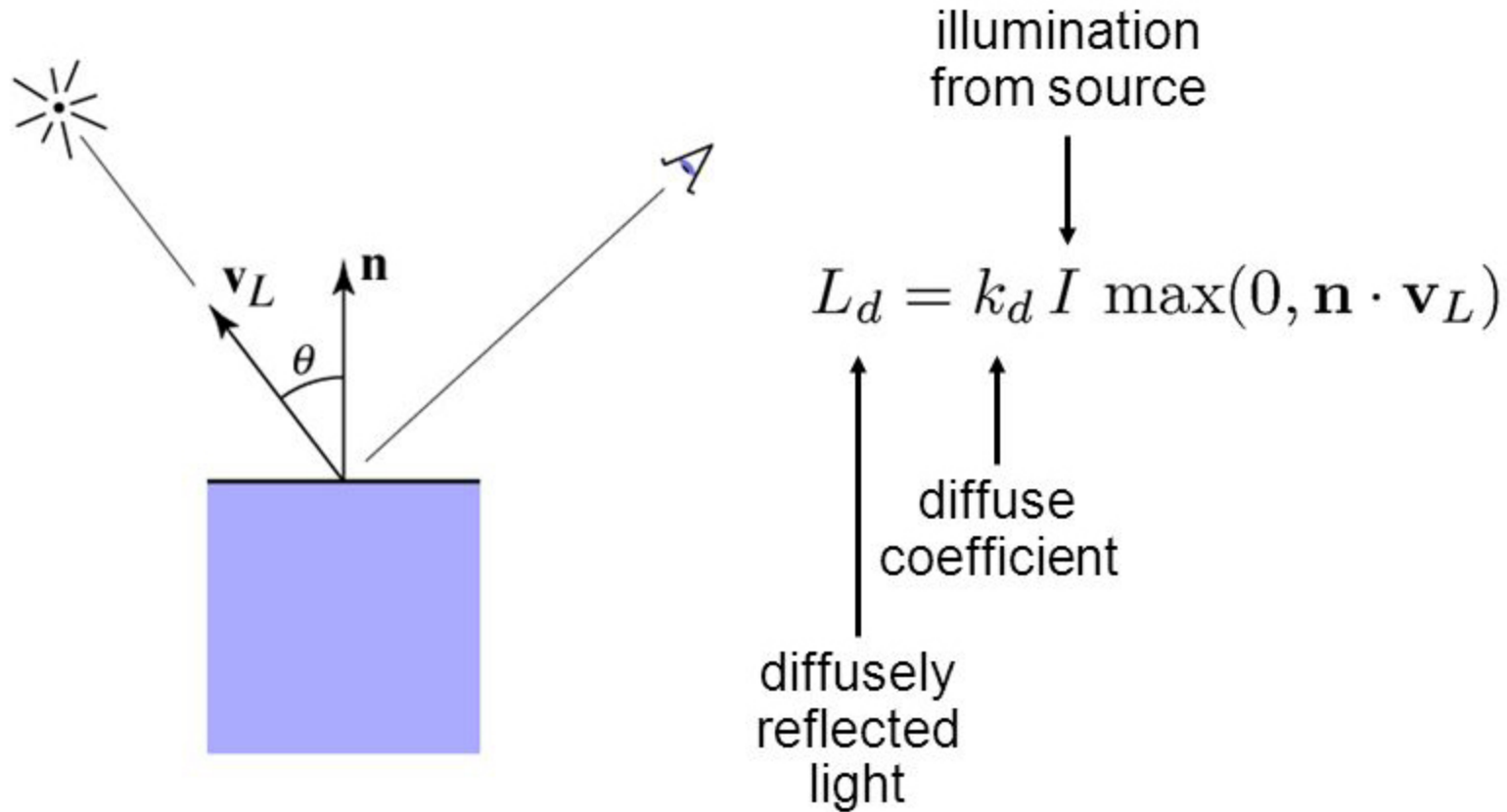
Материалы

- Светоотражающие характеристики объекта задаются его *материалом*.
- Ambient характеристика будет обозначаться как **Ka** (или **Ke**)
- Diffuse - **Kd**
- Specular - **Ks**
- Освещенность в точке p считается как

$$I_p = K_a + K_e + K_d \cdot \text{diffuse}() + K_s \cdot \text{specular}()$$

Diffuse

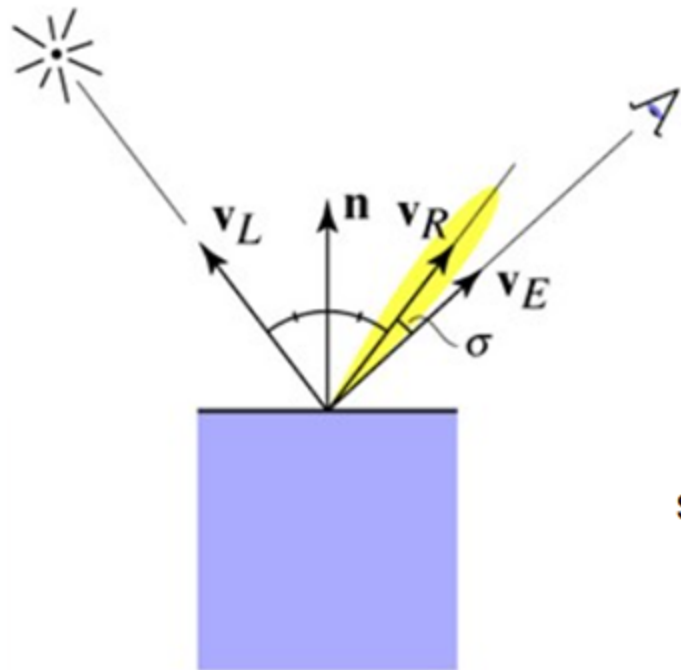
- Shading independent of view direction



Specular

Specular shading (Phong model)

- Intensity depends on view direction
 - bright near mirror configuration



$$\begin{aligned}\mathbf{v}_R &= \mathbf{v}_L + 2((\mathbf{n} \cdot \mathbf{v}_L)\mathbf{n} - \mathbf{v}_L) \\ &= 2(\mathbf{n} \cdot \mathbf{v}_L)\mathbf{n} - \mathbf{v}_L\end{aligned}$$

$$\begin{aligned}L_s &= k_s I \max(0, \cos \sigma)^n \\ &= k_s I \max(0, \mathbf{v}_E \cdot \mathbf{v}_R)^n\end{aligned}$$

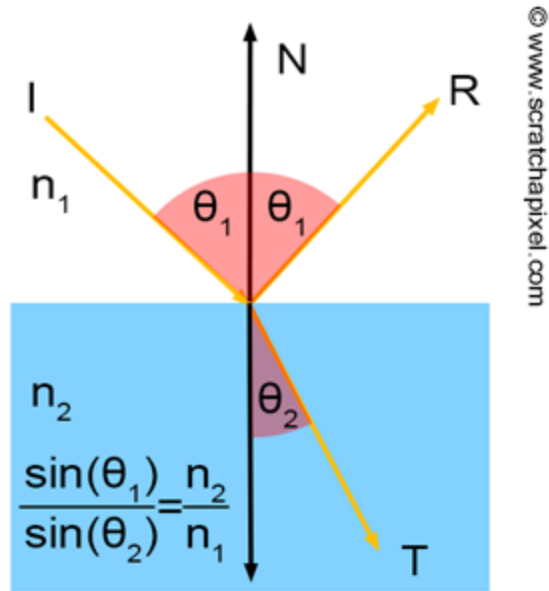
specularly
reflected
light

specular
coefficient

Отражения

- Для некоторых материалов в задании будет задано, что для них необходимо также посчитать отраженный и/или преломленный лучи.
- У нас есть функция, которая трассирует луч в заданном направлении и возвращает получившуюся освещенность (собственно наш основной `Render`).
- Применим ее рекурсивно в направлении отраженного от p луча, получим какую-то интенсивность I' . Трактуем ее как еще один "источник света".
- Глубина рекурсии будет задаваться в `RenderParams` .

Преломления



- Для материала будет задан коэффициент преломления (n_2 на картинке), для остальной среды считаем его равным 1. Материал также характеризуется коэффициентом прозрачности Tr .
- Снова используем рекурсию: трассируем луч в направлении преломления, получаем интенсивность I'' .

Преломления

- В отличие от отраженного луча трактуем не как "источник света", а как отдельную аддитивную прибавку к I_p , т.е. добавляем

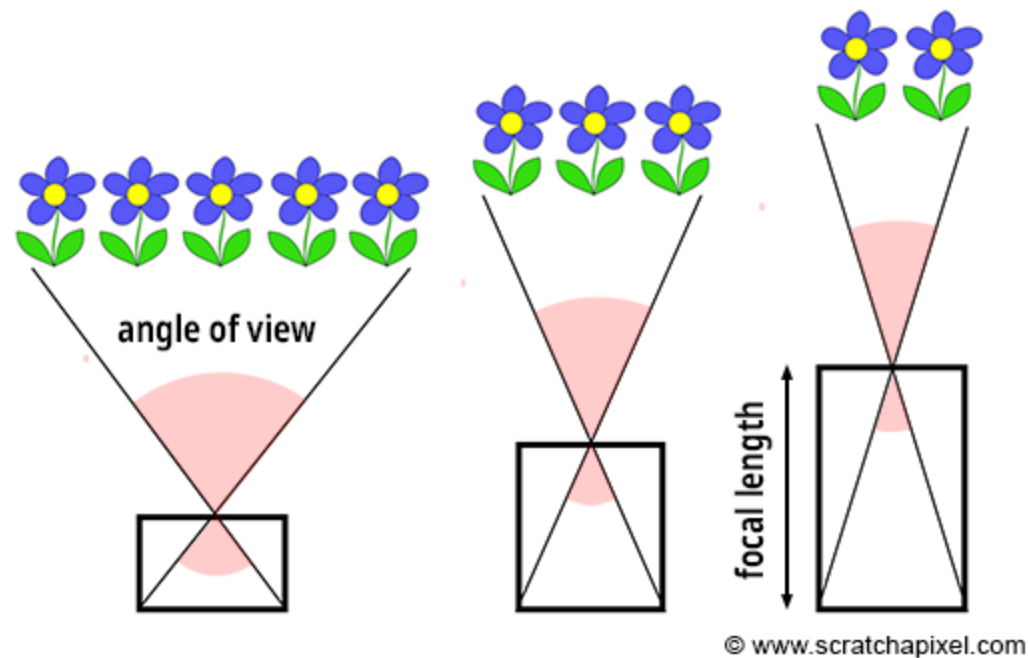
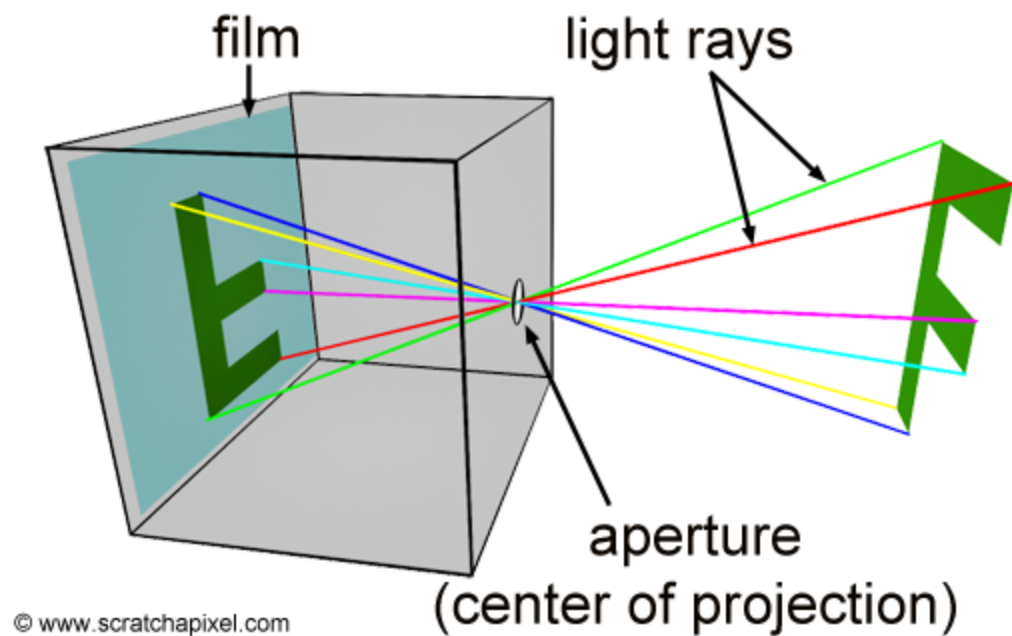
$$I_p+ = Tr \cdot I''$$

- В трассировщике нужно отслеживать находимся мы внутри объекта или снаружи. Когда мы находимся внутри, коэффициенты преломления меняются местами, а $Tr = 1$. Также не нужно считать отраженный луч (даже если для материала указано, что он нужен), если мы находимся внутри объекта.
- По формуле выше нетрудно заметить, что считать преломление не нужно, если $Tr = 0$.

Геометрия

- Обычно все объекты в реальных сценах состоят из полигонов. В нашем случае это будут только треугольники + сферы.
- Нужно уметь пересекать луч с треугольником и сферой, а также брать нормаль к поверхности в точке пересечения.
- Почти все вычисления в задаче векторные, поэтому рекомендуется завести класс `Vector` (3 измерения) и реализовать для него всю нужную арифметику.

Камера и экран

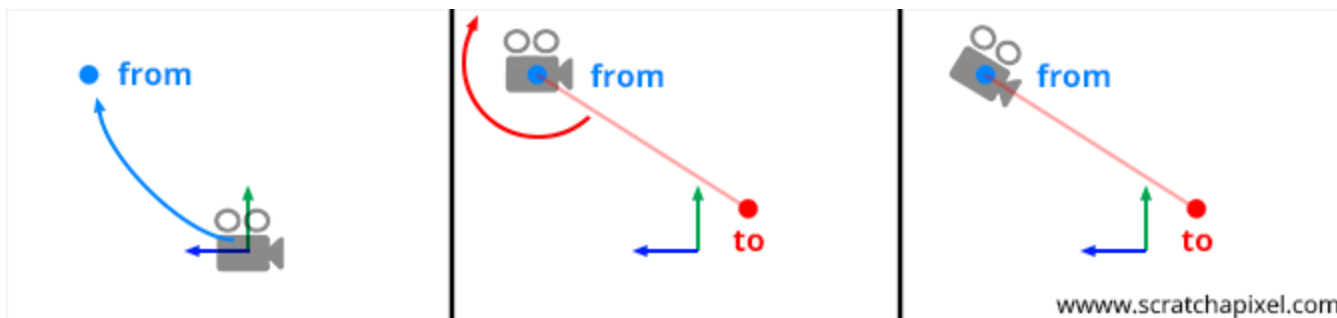


Камера и экран

- Камера по умолчанию находится в начале координат и смотрит в направлении $-z$. На расстоянии 1 (focal length) расположен экран (тот, на который мы смотрим), ось x идет слева направо, y снизу вверх.
- В нашей задаче для камеры можно задавать FOV и размеры экрана.
- Focal length всегда остается равен 1! Это означает, что размеры экрана не задают какую часть сцены мы увидим (это определяется fov и focal length), а определяют размеры "пикселя" в наших координатах
- Лучи нужно трассировать через центры пикселей.

Перемещения камеры

- Вся сцена существует в некоторой системе координат, и когда мы трассируем лучи, эти лучи должны быть в той же системе координат.
- Это можно сделать так: сгенерировать луч в системе координат камеры (как это описано выше), а затем трансформировать его из системы координат камеры в систему координат сцены.
- Мы будем контролировать положение камеры с помощью примитива LookAt: задается положение камеры и точка, в которую она смотрит.



Полезные ссылки

- <https://www.scratchapixel.com/index.php?redirect>, в частности про камеру
 - <https://www.scratchapixel.com/lessons/3d-basic-rendering/ray-tracing-generating-camera-rays/generating-camera-rays>
 - <https://www.scratchapixel.com/lessons/mathematics-physics-for-computer-graphics/lookat-function>
- <https://habr.com/ru/post/436790/>

Возможные проблемные места

- Забыли отнормировать нормали
- От поверхности можно провести 2 нормали, обычно нужна определенная сторона
- При рекурсивном вызове для вычисления отраженного/преломленного луча происходит попадание в ту же точку (нужно не забывать отступать от поверхности по направлению нормали на ϵ)
- Геометрия

Как задается сцена

- Положение объектов в пространстве описано в .obj файлах. Свойства материалов задаются в связанных с ними .mtl файлах.
- Оба типа файлов текстовые и парсятся за 1 проход.

Пример .obj

```
# comment
mtllib materials.mtl

v 0 0 1
v 0 0.1 1.0
v 0.1 0.1 1.1

usemtl triangle
f 1 2 3

v 0.1 0.0 1.1

usemtl example
f -1 -2 -3 -4
```

Пример .mtl

```
# comment  
newmtl triangle  
    Ks 0.5 0 0  
    Ka 0.05 0.3 0  
    Kd 0.1 0.1 0.2  
    Ns 500  
  
newmtl example  
    Ke 7 3 2
```


Дополнительные параметры вершин

```
v 0 0 0
```

```
v 0 1 0
```

```
v 1 0 0
```

```
vt 0 0
```

```
vt 0 1
```

```
vt 1 0
```

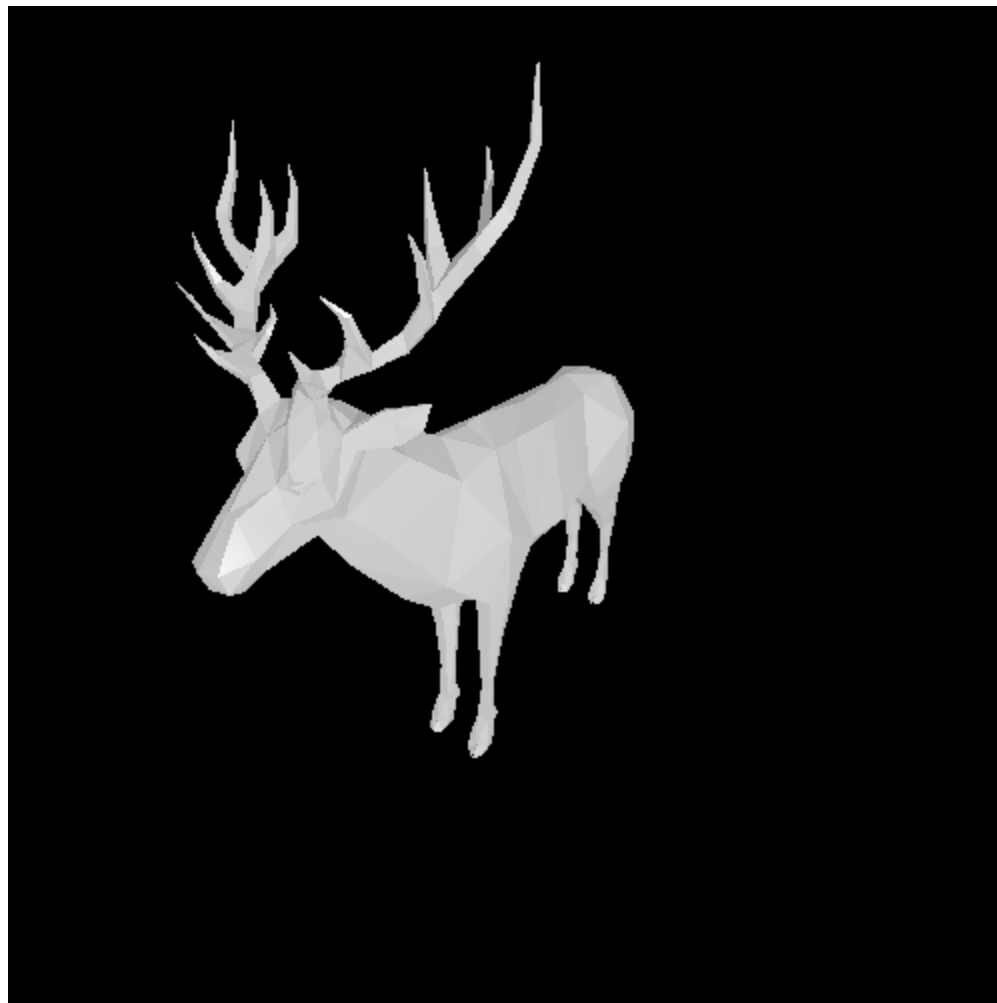
```
vn 0 0 -1
```

```
vn 0 0 2
```

```
vn 0 0 0.5
```

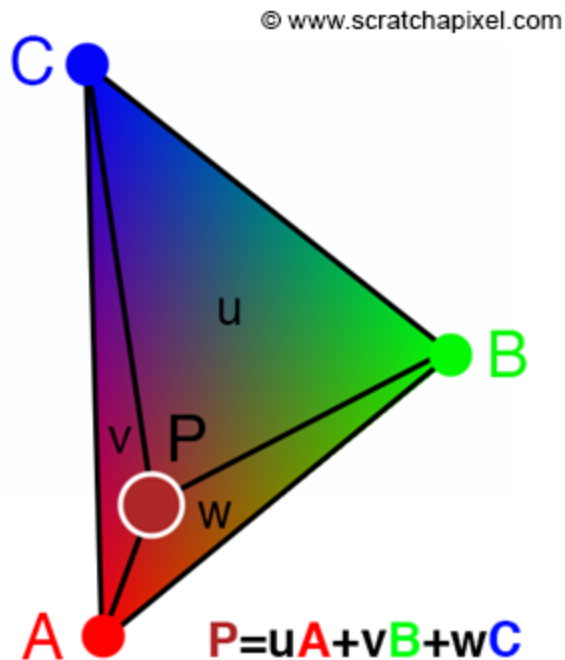
```
f 1/1/1 2/-2/-2 3//3
```

Зачем нужны явные нормали?



Как в итоге посчитать нормаль

- Если пересечение со сферой или для вершин треугольника не заданы vn , то считаем как обычно. Иначе нужно проинтерполировать значения vn вершин треугольника, с которым пересеклись, используя барицентрические координаты.



- Подробнее можно почитать здесь: <https://www.scratchapixel.com/lessons/3d-basic-rendering/ray-tracing-rendering-a-triangle/barycentric-coordinates>

Текстуры (бонус)

- Нужно поддерживать ambient, diffuse и bump текстуры
- Координаты вершин в текстуре задаются в vt. Левый нижний угол текстуры имеет координаты 0 0, а правый верхний 1 1.
- Для получения промежуточных координат используем такое же интерполирование через барицентрические координаты.