

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ  
ФЕДЕРАЦИИ**  
**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ**  
**ВЫСШЕГО ОБРАЗОВАНИЯ**  
**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ  
УНИВЕРСИТЕТ им. В. Г. ШУХОВА»**  
**(БГТУ им. В.Г. Шухова)**



**ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И УПРАВЛЯЮЩИХ СИСТЕМ**

**КУРСОВОЙ ПРОЕКТ**

по дисциплине: Основы программирования

тема: «Разработка графического движка»

Автор работы \_\_\_\_\_ Пахомов Владислав Андреевич ПВ-223  
(подпись)

Руководитель проекта \_\_\_\_\_ Черников Сергей Викторович  
(подпись)

Оценка \_\_\_\_\_

Белгород 2023 г.

# Оглавление

## Введение

### 1 Техники реалистичного рендера

1.1 Трассировка лучей . . . . .

1.2 PBR . . . . .

### 2 Используемые библиотеки и технологии

2.1 Формат хранения сцены . . . . .

2.2 HIP и его расширение HIP RT . . . . .

### 3 Заключение

### 4 Список источников и литературы

## Введение

Большую часть информации человек воспринимает глазами, именно поэтому одним из самых популярных видов контента на сегодняшний день является визуальный контент.

Красочная, яркая и пёстрая или серая, драматичная. За множество лет человечество успело отобразить реальность в графическом формате множество раз в виде картин, фильмов, фотографии.

Компьютеры стали незаменимыми помощниками в создании графического контента. Вычислительные мощности компьютеров, растущие ежегодно, уже позволяют создавать картинку, неотличимую от реальности. Это стало возможно благодаря развитию графических процессоров - отдельному устройству ПК.

Более мощные компьютеры позволяют сегодня использовать более сложные алгоритмы для получения реалистичной картинки, например трассировка лучей и Physically Based Rendering. В последние модели видеокарт добавляются дополнительные ядра, которые способны решать задачи, направленные на рендеринг при помощи данных техник.

Производитель видеокарт предоставляет библиотеки, позволяющие работать с этими ядрами. Для видеокарт от компании AMD такой библиотекой является HIP RT, расширяющая библиотеку для работы с видеокартой HIP.

Объект исследования - разработка графического движка.

Предмет исследования - библиотеки для работы с видеокартами от AMD HIP и HIP RT, техники реалистичного рендера трассировка лучей и Physically Based Rendering.

Цель - разработать графический движок, использующий техники трассировка лучей и Physically Based Rendering и аппаратное обеспечение (видеокарту).

Для достижения поставленной цели необходимо решить следующие задачи:

- Изучить техники реалистичного рендера трассировка лучей и Physically Based Rendering.
- Изучить и применить библиотеки для аппаратного ускорения при ис-

пользовании техник реалистичного рендера.

- Подобрать удобный формат хранения информации о 3D-сцене.
- Разработать программу, генерирующую изображение на основе данных о 3D-сцене.

## 1 Техники реалистичного рендера

### 1.1 Трассировка лучей

В основе трассировки лучей лежит довольно простая идея. Предположим, нам нужно нарисовать картину, но всё что мы можем сделать - это ставить точки и безошибочно определять цвет, куда мы смотрим. Можно разбить холст на квадраты и методично просматривать каждый из них, определяя цвет и ставя точку соответствующего цвета. Таким образом можно получить картину.

Трассировка лучей работает схожим образом. Из точки наблюдения мы будем испускать луч в соответствующем направлении и определять, в какой цвет окрашивать текущий пиксель.

Точка наблюдения - это координаты камеры. Направление испускаемого луча можно определить по следующей формуле:

$x, y$  - координаты текущего обрабатываемого пикселя,

$AR = \frac{Res_W}{Res_H}$  - соотношение сторон,  $Res_W$  - ширина холста,  $Res_H$  - высота холста.

$S_H = \frac{2}{1+AR}$ ,  $S_W = 2 - S_H$  - стороны прямоугольника, подобного холсту, причём  $S_H + S_W = 2$ .

$D = ((x/Res_W) \cdot S_W - \frac{S_W}{2}, (y/Res_H) \cdot S_H - \frac{S_H}{2}, (-S_W/2)/\tan(FOV/2))$ , где FOV - вертикальный обзор камеры. Дополнительно вектор D можно умножить на матрицу вращения для того, чтобы повернуть обозревателя.

Будем находить, пересёкся ли луч с каким-либо объектом, и если пересёкся, ставить точку его цвета. Иначе - чёрную.

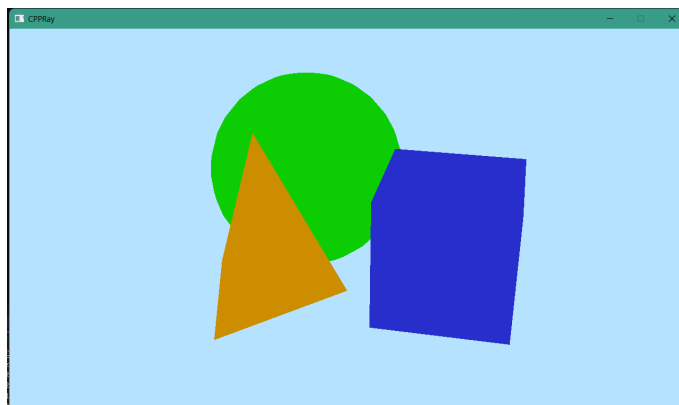


Рисунок 1: Пересечение луча и фигуры

Хотелось бы немного разнообразить сцену - да будет свет! Введём направленные источники света, иначе говоря - солнца. Солнечный свет имеет направление, и следовательно освещать объекты будет по-разному. Чем больше угол между солнечным лучом и нормалью вершины, тем меньше солнца он будет получать.

$Color = BaseColor \cdot L_C \cdot L_I \cdot \cos(-N, L_D)$ , где  $BaseColor$  - цвет объекта,  $L_C$  - цвет света,  $L_I$  - интенсивность света,  $N$  - нормаль объекта,  $L_D$  - направление света.

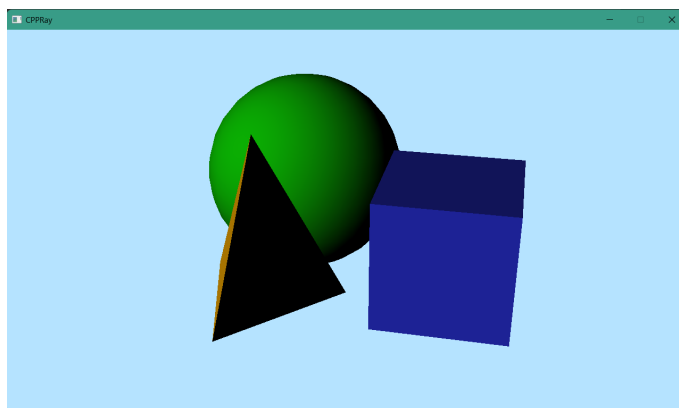


Рисунок 2: Сцена с источником света

Сцена стала немного интересней, однако она всё ещё довольно странная - предметы не отбрасывают тень. Для того, чтобы понять, отбрасывает ли объект тень, можем испустить луч от точки пересечения в противоположном направлении свету. Если мы найдём хоть один объект, который пересекает луч, то значит в данной точке будет тень. Иначе - точка освещена.

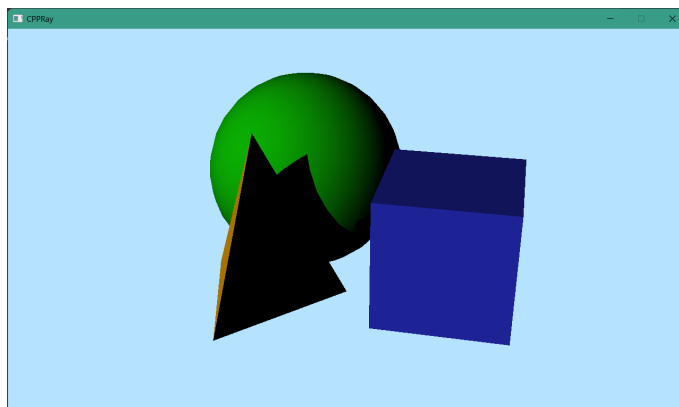


Рисунок 3: Добавление тени

Тетраэдр начал отбрасывать тень на сферу. Можно также добавить другие источники света - точечный свет и лампы. Основным отличием от солнца у этих источников света является затухание. С расстоянием сила света будет становиться меньше. Затухание можно рассчитать по следующей формуле:

$Att = \max(\min(1 - \frac{Distance^4}{L_R}, 1), Distance^2)$ , где Distance - расстояние между точкой на объекте и источником света,  $L_R$  - радиус света.

Точечный свет находится в одной точке и испускает свет во все стороны, формула для получения света будет следующей:

$$Color = BaseColor \cdot L_C \cdot L_I \cdot \cos(-N, L_D) \cdot Att$$

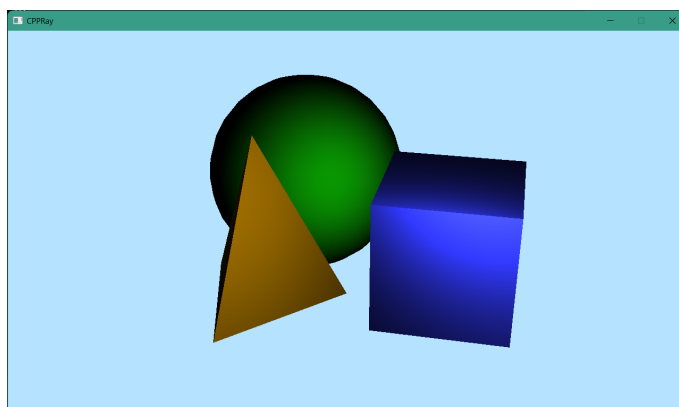


Рисунок 4: Точечный свет

Для определения тени будем испускать луч из источника света в направлении к рассматриваемой в данный момент точке. Если ближайшее пересечение с объектом - пересечение с искомым объектом, то он освещён. Иначе - оставляем тень.

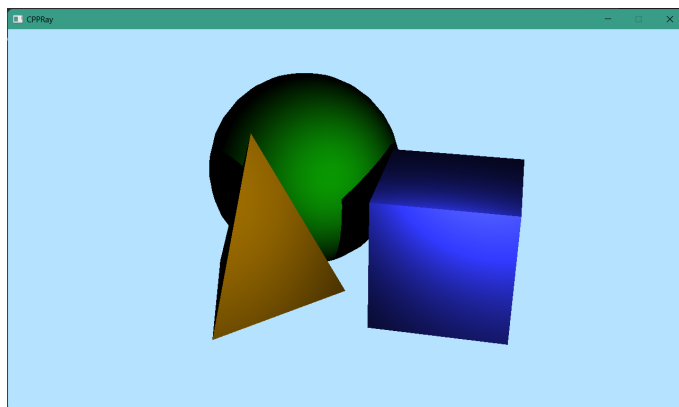


Рисунок 5: Точечный свет с тенью

В источнике света "лампа" появляются углы внутреннего и внешнего конусов. Свет, находящийся во внутреннем конусе, имеет максимальную интенсивность, между внешним и внутренним конусом затихает, и вне внешнего конуса света нет.

Для вычисления интенсивности света используется следующая функция:

$$k = \begin{cases} 1 & \text{angle} < ICA \\ 0 & \text{angle} > OCA \\ \frac{\text{angle}}{ICA - OCA} + 1 - \frac{ICA}{ICA - OCA} & \text{else} \end{cases}$$

где  $\text{angle}$  - угол между направлением лампы и вектором от источника света до точки на объекте,  $ICA$  - радиус внутреннего конуса,  $OCA$  - радиус внешнего конуса.

Просчёт наличия тени будет выполняться аналогично с точечным светом.

TODO: Вставить пикчу сюды

## 1.2 PBR

До сих пор мы рассматривали взаимодействие света и объекта упрощённо, однако в реальности такого не бывает.

Техника Physically Based Rendering позволяет определить, как свет будет взаимодействовать с объектом в зависимости от его физических свойств - шероховатости и металличности.

## **2 Используемые библиотеки и технологии**

### **2.1 Формат хранения сцены**

Перед тем как начать рендер 3D-сцены, необходимо получить информацию о ней. Можно было "хардкодить" информацию о ней прямо в коде, однако полученная программа будет крайне негибка: требовать постоянной рекомпиляции, работы с программистом.

Информацию о сцене можно получать из файла. Выбор пал на несколько форматов хранения информации о 3D-сценах:

- STL - простой формат файла, однако не подходящий для реалистичного рендера, он содержит только информацию о форме объектов.
- FBX - популярный формат файла, разрабатываемый в Autodesk. Формат тоже не подходит, так как не содержит информацию для выбранной техники Physically Based Rendering.
- COLLADA - формат, основанный на XML и разработанный для передачи информации между 3D приложениями. Управляется Khronos Group. Формат также не поддерживает Physically Based Rendering.
- glTF - формат, основанный на JSON, расширяемый, легкопонижаемый. Используется в веб-технологиях. Поддерживает все необходимые данные для выбранных техник.

Для хранения сцены был выбран формат glTF. Для обработки glTF-файла была использована библиотека tinyglTF.

### **2.2 HIP и его расширение HIP RT**

Основное преимущество графического процессора, или видеокарты, заключается в возможности выполнять множество процессов одновременно.

## **3 Заключение**

## **4 Список источников и литературы**

<https://gpuopen.com/hiprt/> <https://github.com/syoyo/tinyglTF> <https://habr.com/en/articles/3>