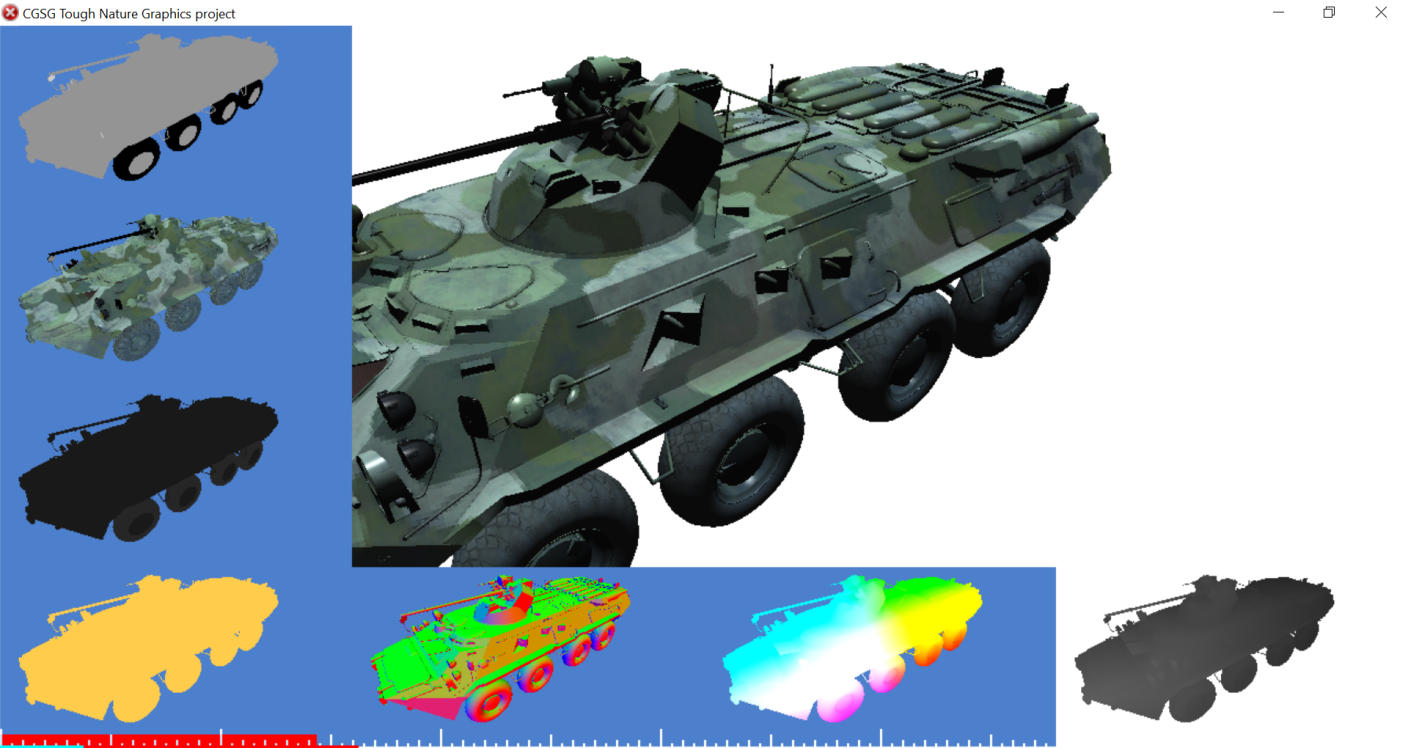
**Санкт-Петербургский губернаторский  
физико-математический лицей № 30**

**Проектирование системы визуализации трехмерных сцен в реальном времени с использованием современных средств графических процессоров**



Авторы:

* ***Двас Павел Григорьевич, 08-2 класс***
* ***Марков Максим Александрович, 08-2 класс***
* ***Кураленок Святослав Игоревич, 08-4 класс***
* ***Вашпанов Александр Викторович, 09-3 класс***
* ***Кононов Святослав Игоревич, 09-3 класс***
* ***Шайда Андрей Игоревич, 09-4 класс***
* ***Кузьмин Илья Дмитриевич, 09-5 класс***
* ***Хлевной Евгений Александрович, 09-5 класс***
* ***Копейкина Софья Евгеньевна, 10-5 класс***
* ***Фабрика Артем Александрович, 10-5 класс***
* ***Гирвиц Александр Михайлович, 10-6 класс***
* ***Дмитриева Василиса Валерьевна, 10-6 класс***

Научный руководитель:

*Галинский Виталий Александрович,*

преподаватель информатики и программирования физико-математического лицея № 30,

руководитель группы компьютерной графики, зам. директора по ИТ

Государственное бюджетное общеобразовательное учреждение "Санкт-Петербургский губернаторский физико-математический лицей № 30"

2018-2019

# Введение

# Целью проекта было построение полноценной системы анимации трехмерных сцен. Система должна содержать полноценный функционал для создания и визуализации пространственных геометрических объектов, построение реалистичной модели освещения множественными источниками света, выполнять синхронизацию анимации по времени. Поставленная перед авторами задача также включает в себя оптимизацию визуализации за счет переноса процесса вывода на видеокарту.

# Разработка проекта включила в себя исследование средств вывода современных видеокарт, построение многоуровневой системы визуализации с возможностью гибкого управления процессом отображения графической информации.

# Реализация проекта выполнена на языке программирования Си, с использованием графической библиотеки OpenGL, для оптимизации вывода используется язык шейдеров GLSL.

# Описание проекта

# Проект состоит из нескольких взаимосвязанных частей. Это:

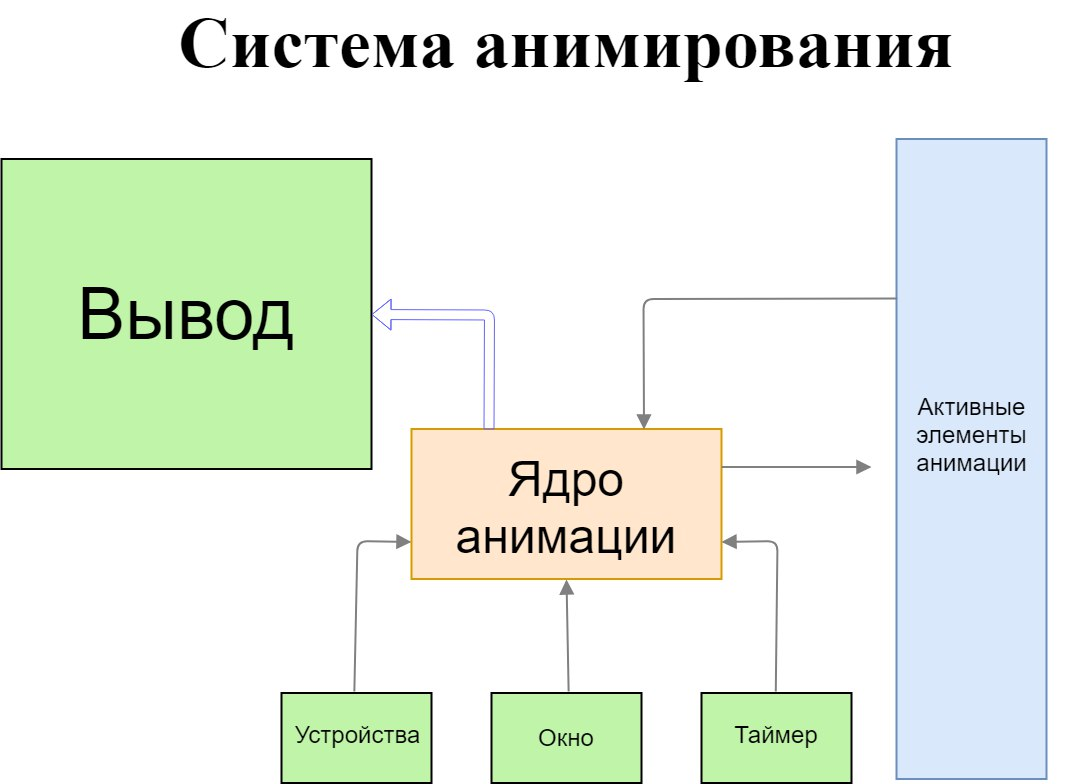
* ***подсистема анимации***, отвечает за основной процесс вывода пространственных объектов, взаимодействие с устройствами ввода, синхронизацию по времени;
* ***подсистема вывода***, отвечает за весь поток визуализации геометрической информации, создание и управления ресурсами, такими, как материалами (включают в себя параметры освещения и текстуры), источниками света, шейдерами (микропрограммы, выполняющиеся на видеокарте);
* ***вспомогательные подсистемы*** — математическая библиотека, построение "топологий" (вспомогательных геометрических данных, информация из которых передается на видеокарту), подсистема воспроизведения пространственных звуков, единая подсистема хранения и управления всеми ресурсами программы.

## Подсистема анимации

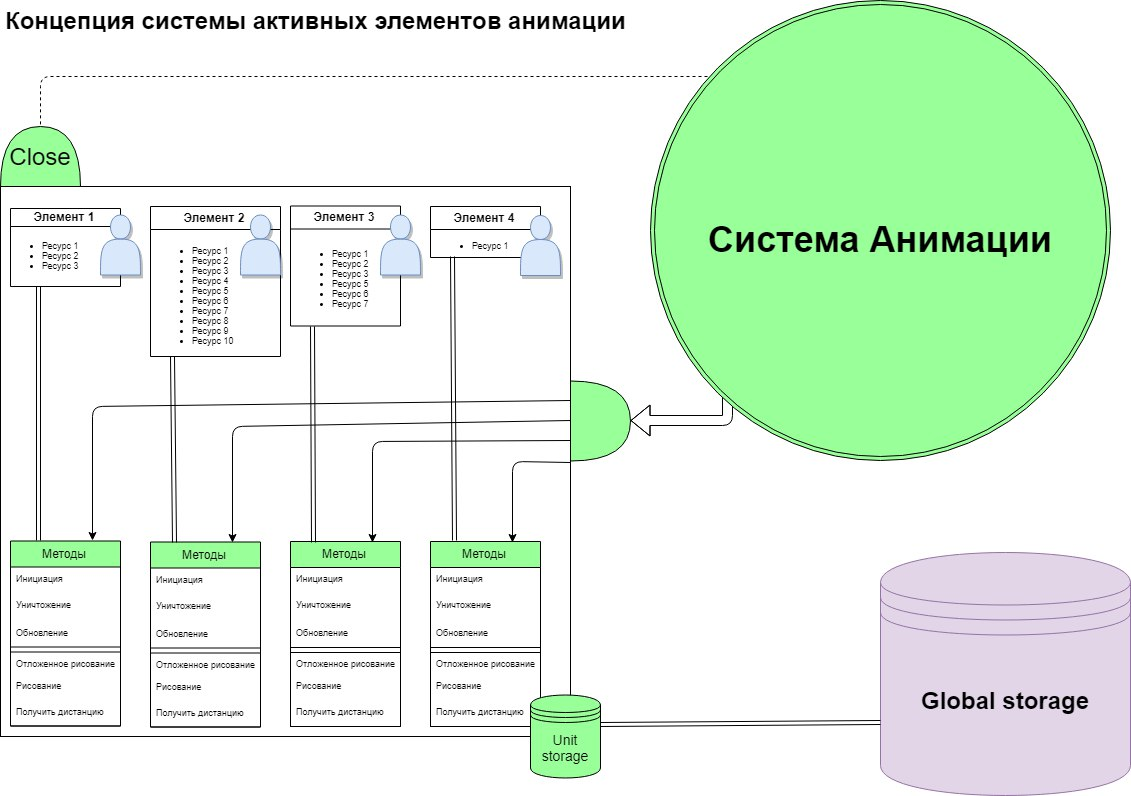
# Система Анимации — главная связующая часть проекта. Создание системы анимации нацелено на централизацию процессов обработки данных в проекте.

# Посредством централизации системы визуализации осуществляется синхронизированная работа процессов анимации, таких как обработка сообщений от оконной системы, взаимодействие с активными элементами анимации, просчёт и вывод сцен на экран. Деление основного цикла анимирования на подсистемы обеспечивает абстрагирование системы анимации от внутренних ресурсов и процессов каждого этапа анимирования. Доступ к любой системной информации из любого блока программы осуществляется через контекст анимации — списка доступных переменных и функций, "сердца" проекта. Благодаря объединению доступных ресурсов подсистем и непрямому доступу к данным обеспечивается универсальность использования объектами системы информации. Также в систему анимации входит метод отложенного вывода активных элементов системы, который связывает внутренние процессы вывода и отложенные действия с активными элементами анимации.

**Схема работы системы анимации.**



# Система активных элементов анимации — блок, который порождает рисование. Активные элементы анимации отличает уникальность внутри своей подсистемы и универсальность для анимации. Каждому элементу анимации присуще 6 методов, соответственно инициация, обновление, прорисовка, отложенная прорисовка, получение дистанции до наблюдателя и уничтожение. Обращение к данным функциям доступно анимации, внутренние ресурсы элементов от ядра скрыты. Также для корректной обработки отложенного вывода существует регистрация на отложенный вывод, которую надо проходить на каждом кадре. Регистрационный лист только создаёт позиции для регистрации, но никогда до прекращения работы системы анимации не удаляет.



## Система ввода информации

# Безусловно, система опроса устройств ввода информации является одной из самых важных в проекте. С ее помощью есть возможность правильно настроить управление камерой, изменять некоторые параметры по желанию пользователя посредством нажатия определенного сочетания клавиш, взаимодействовать с выводимыми моделями и сценами, а также добавлять многие другие функции, позволяющие улучшить работу программы и сделать ее более понятной в использовании.

# Помимо опроса стандартных устройств ввода, а именно мыши и клавиатуры, в проекте существует поддержка игровых джойстиков, с помощью которых можно улучшить взаимодействие с камерой, добавить новые функциональные клавиши и оси, и многое другое.

# Стоит отметить, что на ранней стадии запуска программы данная система запоминает начальное состояние клавиш и осей, таким образом, позволяя использовать уже нажатые клавиши.

**Схема работы системы ввода информации**

**Использование системы ввода информации в проекте**

## Подсистема вывода

# Подсистема вывода отвечает за "rendering". Rendering — это процесс создания финального изображения или последовательности из изображений на основе двухмерных или трехмерных данных.

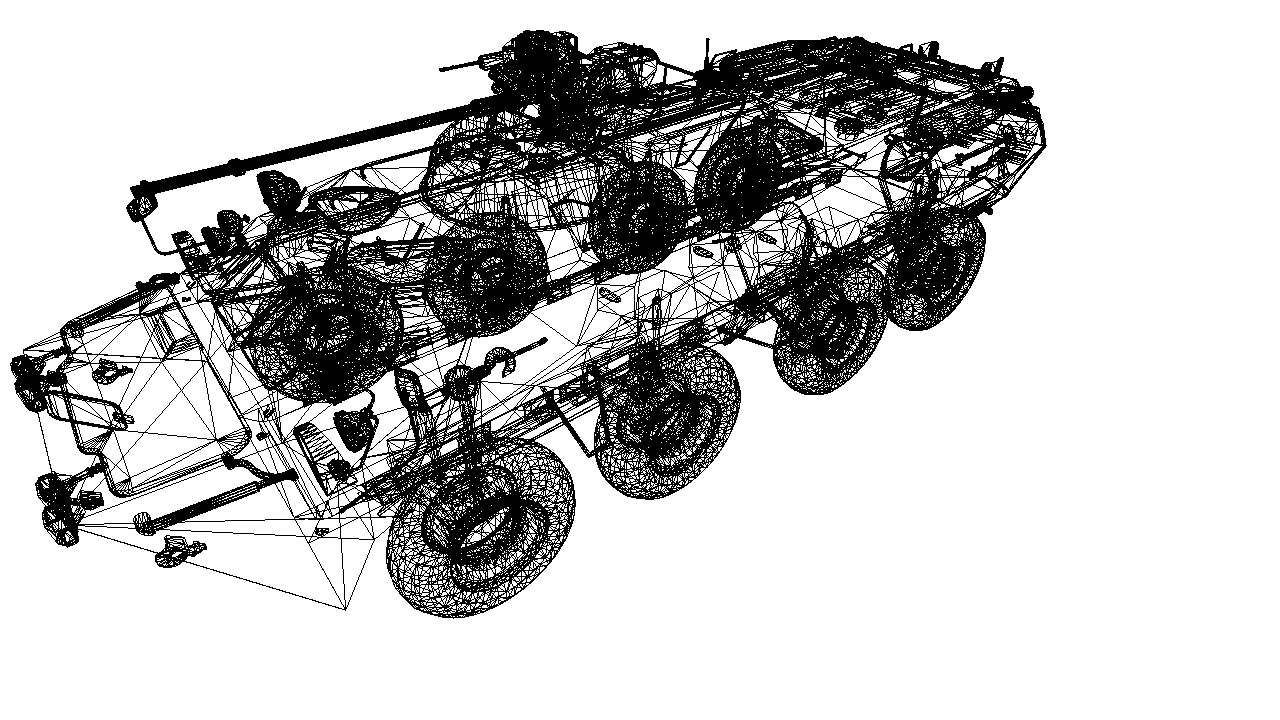
# В проекте вывод состоит из нескольких этапов. Первым этапом идёт инициализация библиотеки OpenGL, подключается набор расширений и новых функций, для работы с видеокартой. Далее идёт последовательная инициализация модулей: топология, текстуры и материалы, подключения шейдеров, освещения и теней и т.д., а также создаются объекты кадра экрана. Они необходимы для построения выводимой информации поэтапно для выполнения процесса освещения.

# Этапы построения (конвейер вывода):

* каждый примитив выводится на экран, при этом вместо вывода непосредственно объекта в объект кадра экрана в разные слои выводятся параметры освещения
* запускается фаза освещения — каждый источник света "выводит себя" на экран, при этом информацию о коэффициентах освещения источник берет из плоскостей объекта кадра экрана
* результат освещения записывается в результирующий объект кадра экрана для последующего вывода на экран
* вторичный объект кадра экрана выводится на экран, при этом существует возможность вывода дополнительной информации (например, статистики и общих параметров) в отложенном режиме на экран.

Примеры плоскостей объекта кадра экрана (построения выполнены в описываемом проекте):

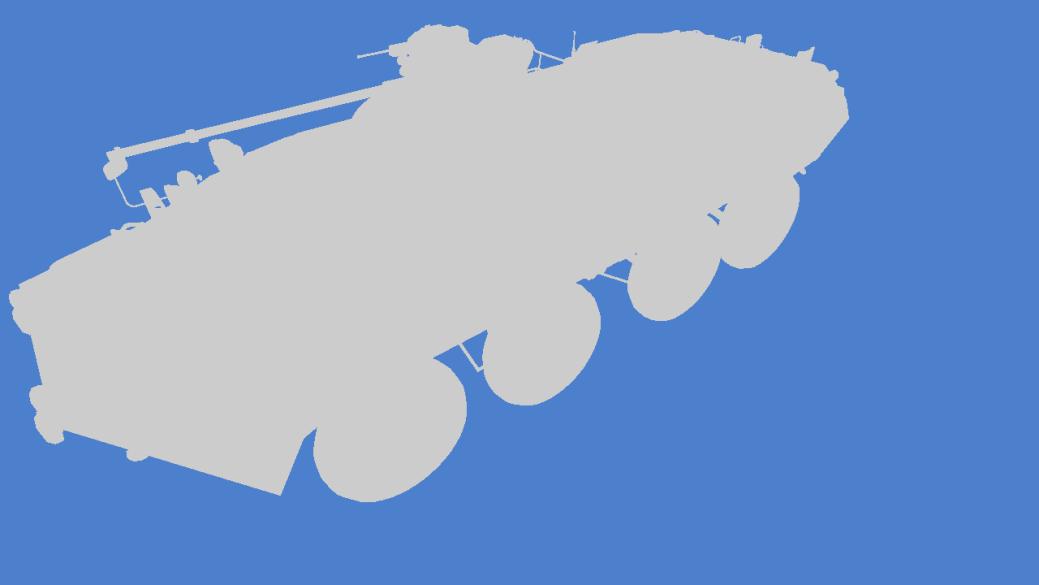
Каркасная модель объекта построения:



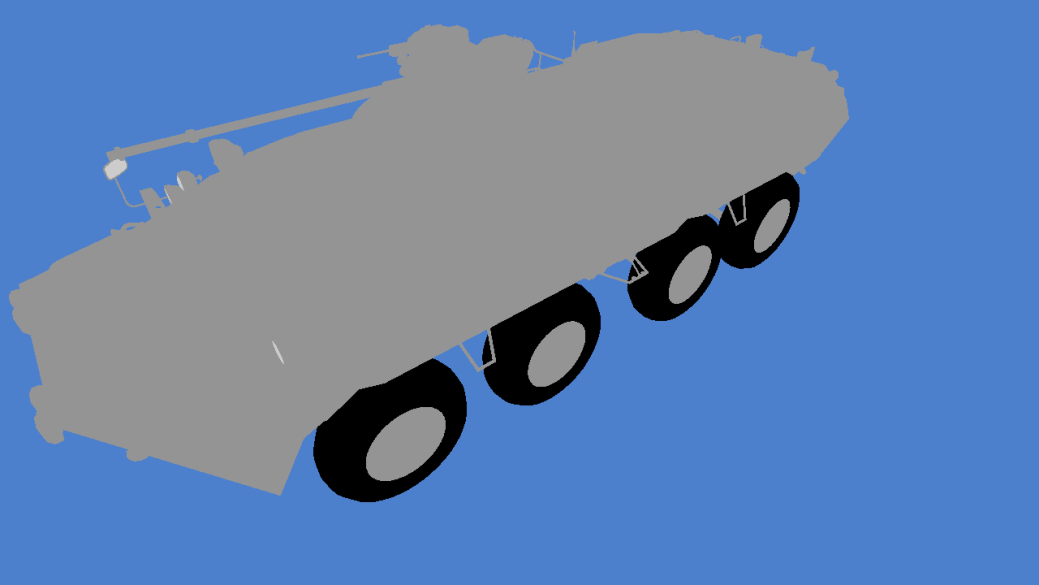
Плоскость цвета построения (базовый цвет):



Плоскость коэффициента зеркальной составляющей (в обозначениях модели освещения Ks):



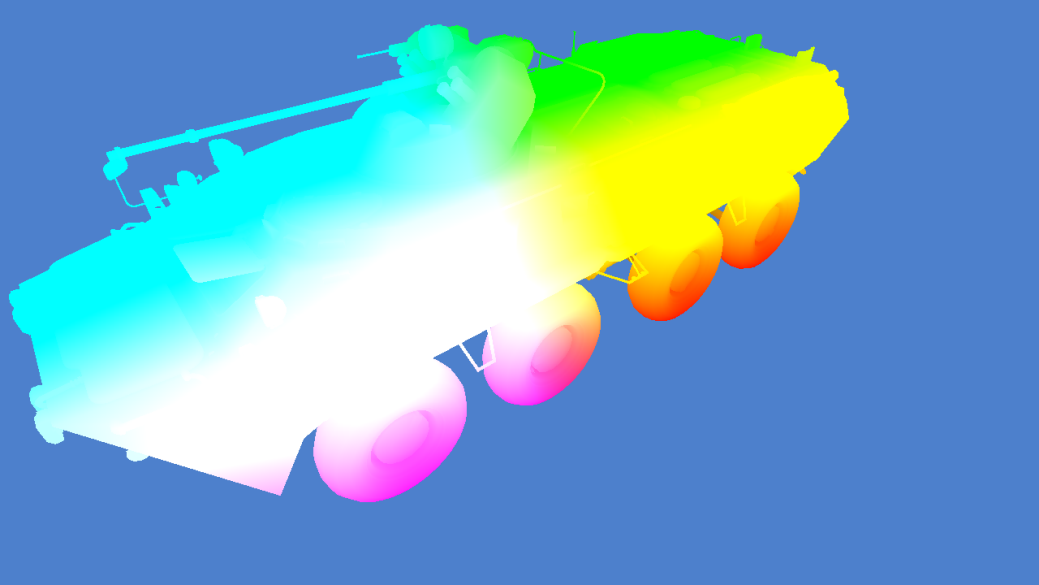
Плоскость коэффициента диффузной составляющей (в обозначениях модели освещения Kd):



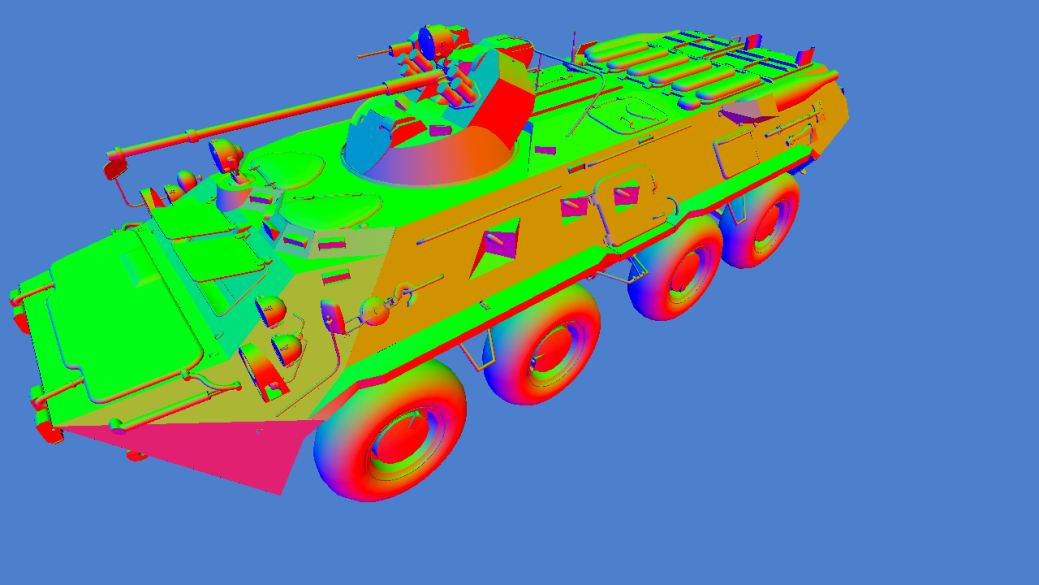
Плоскость коэффициента диффузной составляющей с текстурой:



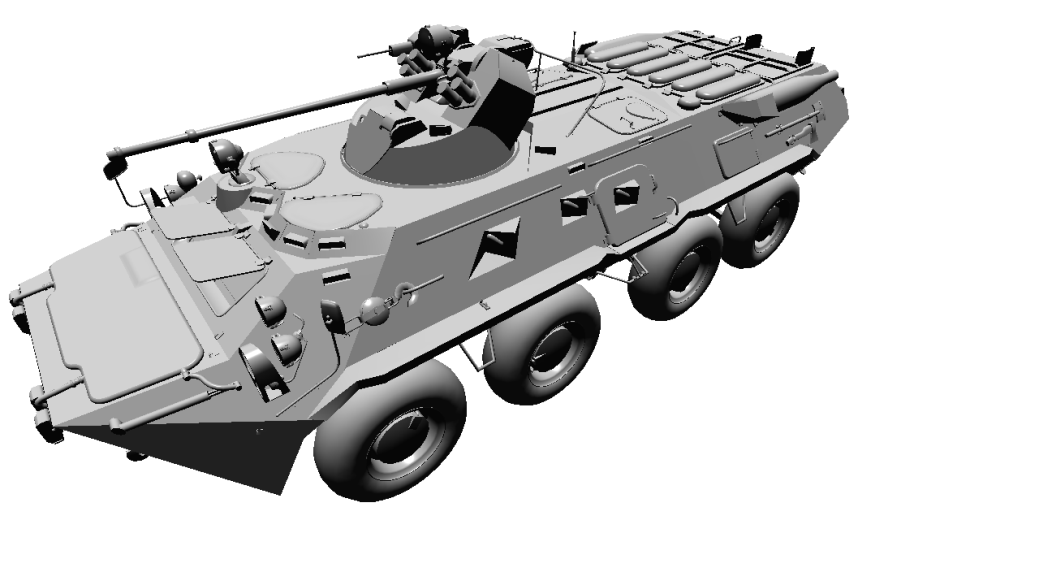
Плоскость позиции точек объекта в пространстве:



Плоскость нормали в точках поверхности объекта:



Результат применения базовой модели освещения:



Результат применения полной модели освещения:



Пример полного построения с отладочным выводом плоскостей объекта экрана (снизу справа отображается буфер глубины, использующийся для удаления невидимых поверхностей):

# TNG_Trg_20190407_140153_b00.bmp

### Примитивы:

# Примитивы — объекты, выводящиеся на экран. В них содержатся буфера, массивы и количество вершин и индексов. В каждой вершине находится позиция её и её текстуры, её нормаль и цвет. К ним привязываются материалы с компонентами отражения, рассеивания, окружения, их прозрачность и блик. Также у примитивов есть базовые типы: точка, треугольник, линия, линия треугольников и так далее.

### Текстуры:

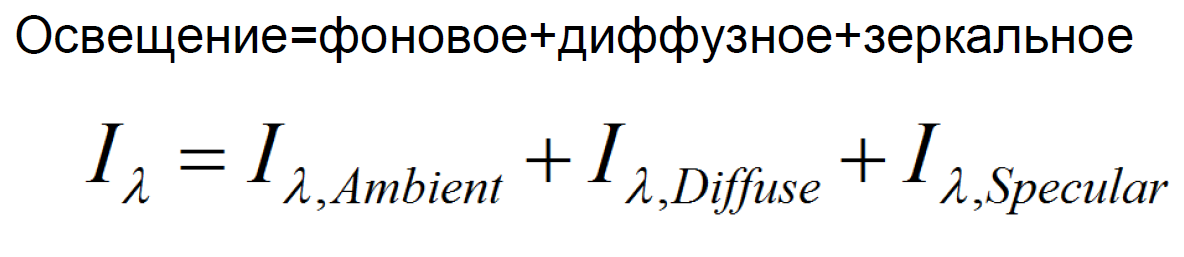
# Текстуры — двухмерные изображения, которые будут использованы примитивом во время освещения, как множитель коэффициента диффузного освещения или же в иных целях (буфера, цветовые "аттачменты" и т.д.)

### Материалы:

# Материалы — набор параметров, задающих цветовые качества объекта. В каждой структуре материала содержатся такие параметры: коэффициент фонового освещения, коэффициент диффузного освещения, коэффициент рассеянного освещения, коэффициент Фонга, прозрачность материала, шейдер, который должен обрабатывать материал, а также текстуры материала. Материалы создаются посредством строкового описания на разработанном языке описания. Полученную строку, содержащую параметры, модуль материалов разбирает с помощью модуля сканера.

### Освещение и затенение:

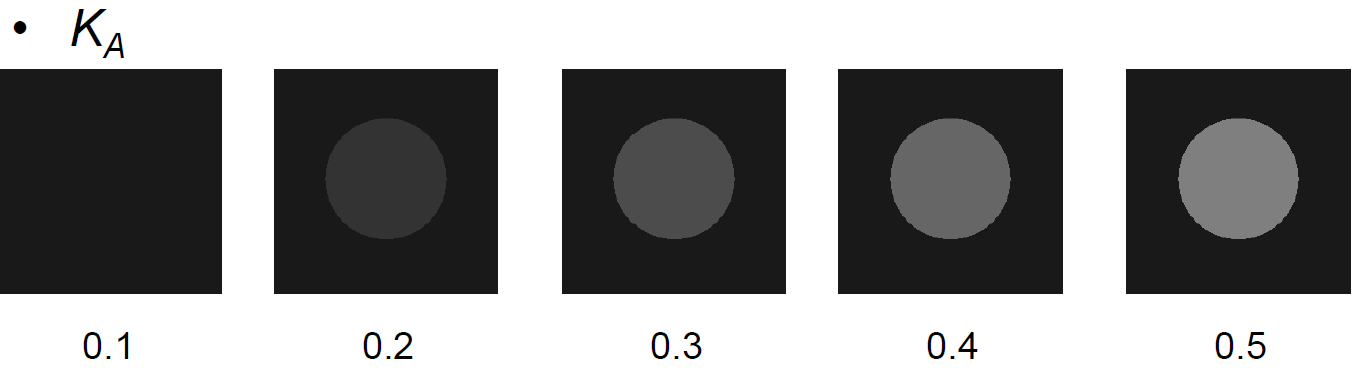
Освещение и тени улучшают реалистичность видимого изображения. Перед авторами была поставлена задача реализовать несколько видов ламп и осветительные модели. Основная осветительная модель – модель Фонга. Вот её алгоритм:



Общее освещение объекта складывается из фонового, диффузного и зеркального освещения.

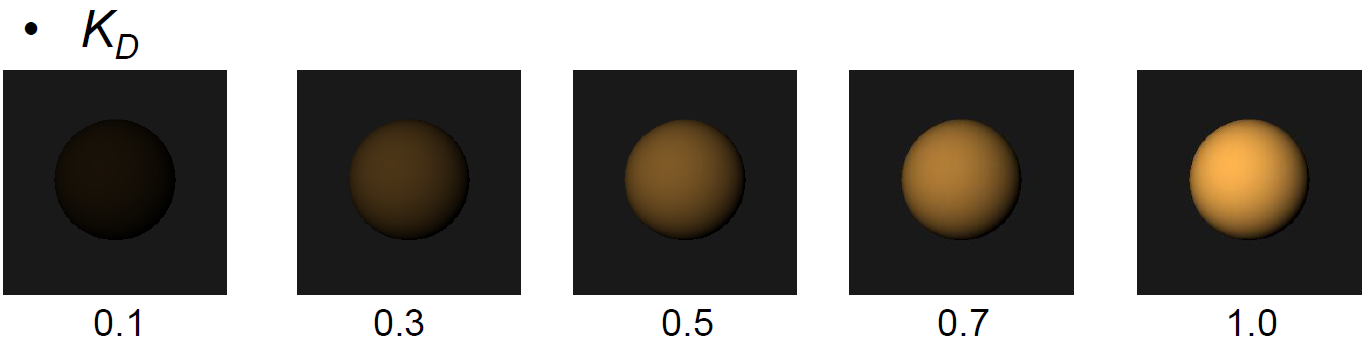
* Фоновое освещение (ambient):

Основным определяющим фактором является коэффициент фонового освещения, в дальнейшем будем сокращать его до KA. Фоновое освещение является перемножением цвета сцены, KА и изначального цвета объекта (если он был).



* Диффузное освещение (diffuse reflection):

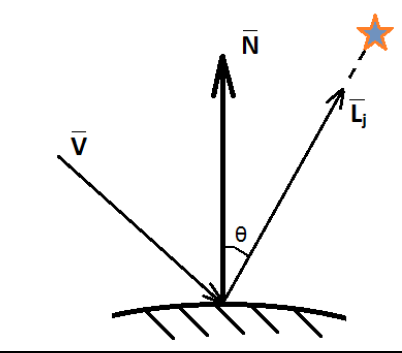
Диффузное освещение является основной частью осветительной модели Фонга. Диффузное освещение для объекта определяется его коэффициентом диффузного освещения, будем сокращать до KD.



Диффузное освещение является перемножением KD, цвета объекта(если он есть), цветом всех ламп, которые его освещают и косинусом между нормалью к поверхности и направлением на источник света(см. рисунок).

На этом рисунке обозначены:

- источник света, – направление на источник света,  -нормаль к поверхности,   – направление на наблюдателя



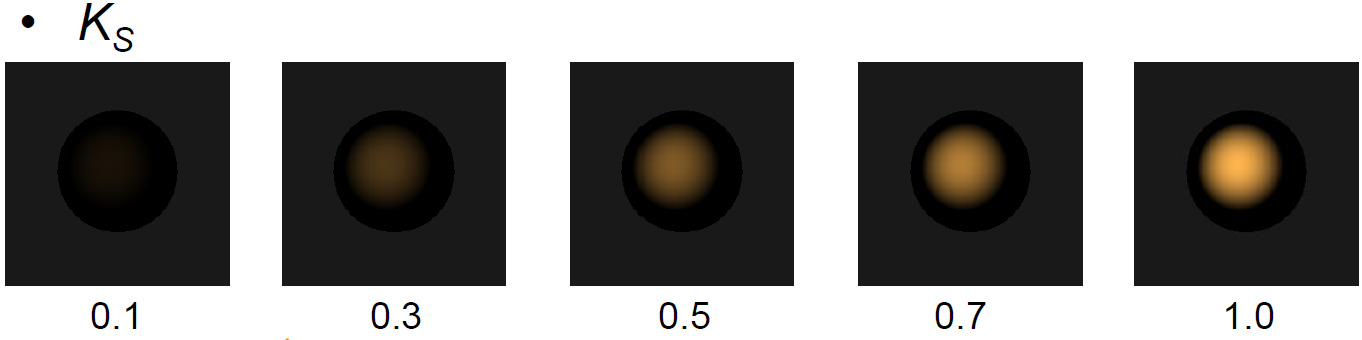


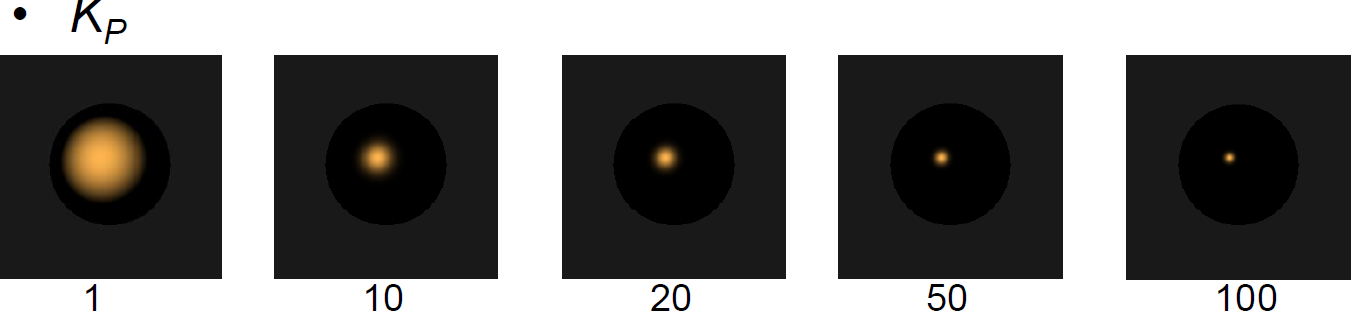
Переход от первого уравнения ко второму выполнен, для того, чтобы показать, как вычислить , зная нормаль и направление на источник света.

* Зеркальное освещение(specular reflection)

Зеркальное освещение используется для имитации бликов на объекте. Оно

определяется коэффициентом зеркального освещения, обозначаем KS и коэффициентом Фонга, обозначаем KP.

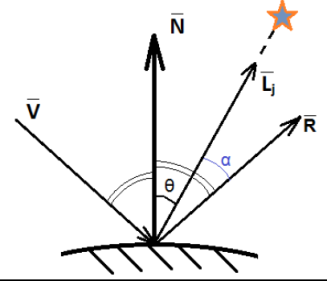




Зеркальное освещение является произведением KS, цвета всех ламп, которые его освещают и косинуса угла между направлением на источник света и вектором отражения, в степени KP.(см. рисунок)

На этом рисунке обозначены:

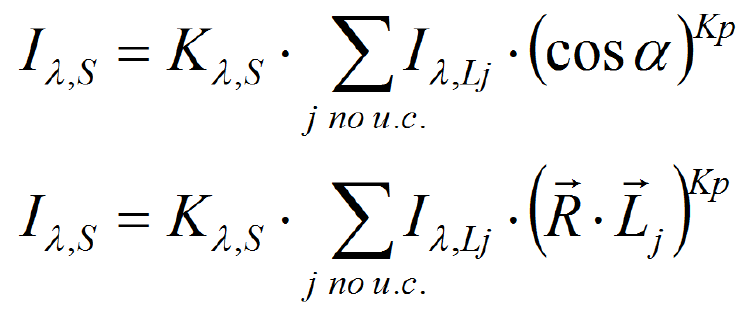
- источник света, – направление на источник света,  -нормаль к поверхности,   – направление на наблюдателя,   – вектор отражения.



Разберемся откуда получается вектор отражения.

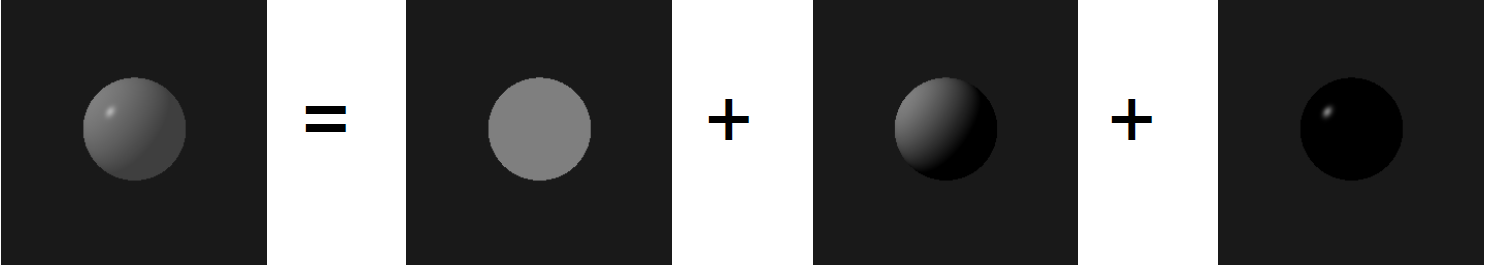
*–* расстояние между точкой, лежащей на окружности единичного радиуса, и принадлежащей , и , где построен два раза от точки пересечения с .





Переход от первого уравнения ко второму выполнен, для того, чтобы показать, как вычислить , зная вектор чистого отражения и направление на источник света.

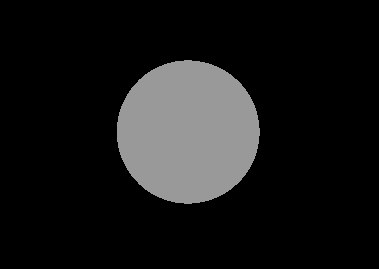
И в итоге получаем достаточно реалистичное изображение.



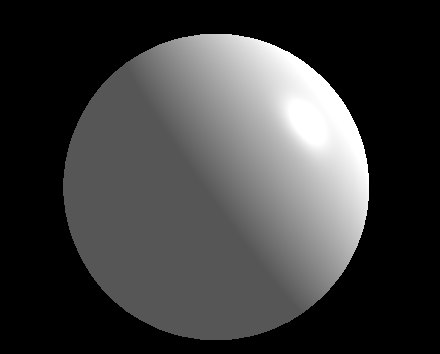
Перейдем к различным видам ламп.

В нашем проекте поддерживается четыре вида ламп:

* Фоновый свет (ambient),



* Точечный источник света (point light),



* Направленный источник света (directional light),
* Источник света в виде конуса (spot light).

### Топология:

Топология геометрических объектов в пространстве. Была поставлена задача создания простых геометрических объектов с использованием математических формул. Использование топологии значительно упрощает построение 3D сцен. Благодаря своей структуре мы храним количество вершин и их характеристики: такие как текстурные координаты, цвет точки, позиция, нормаль к поверхности, тангент и битангент. Из-за этого мы можем с лёгкостью передавать вершины сразу же на видеокарту, что значительно ускоряет процесс рендера.

Прикладных задач у топологии очень много. Одной из них является генерация ландшафта. Используются технология построения по карте высот.

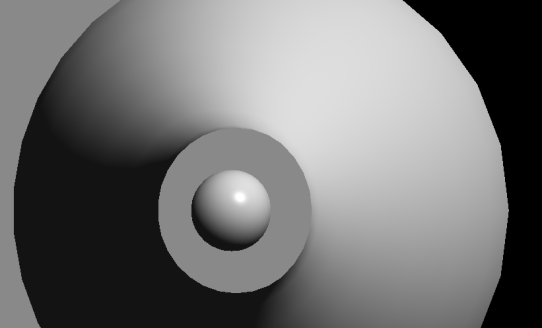
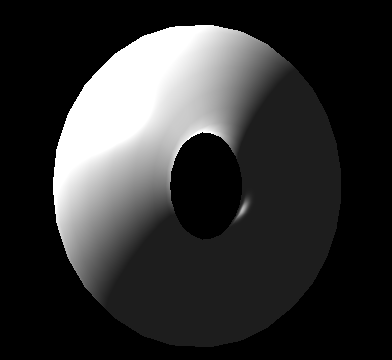
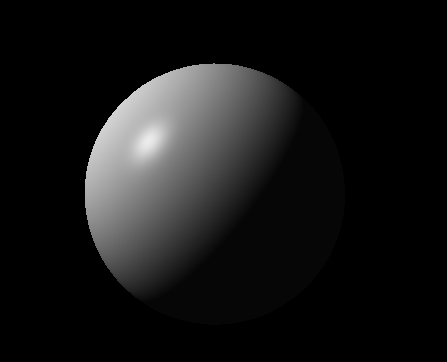
Рассмотрим сферу: x = r \* cos(θ) \* sin(ϕ); y = r \* sin(θ) \* sin(ϕ); z = r \* cos(θ)

где θ - это азимутальная координата от 0 до 2pi (долгота), ϕ - полярная координата от 0 до pi,

а r – радиус сферы.

Рассмотрим тор: x = (R + r \* cos(θ)) \* cos(ϕ); y = (R + r \* cos(θ)) \* sin(ϕ); z = r \* sin(θ);

где θ - это азимутальная координата от 0 до 2pi (долгота), ϕ - полярная координата от 0 до pi, r - внутренний радиус, а R - внешний радиус.



### Аудио:

Подсистема звука необходима для добавления в сцену эффекта трехмерного свучания. Подсистема позволяет выполнять воспроизведение/остановку проигрывания, управление громкостью проигрывания, зависимость громкости проигрывания от расстояния до камеры, изменение позиции источника звука, изменение дистанции слышимости проигрывания. Для взаимодействия с аудиоаппаратурой используется библиотека OpenAL.

# Заключение

Разработанный проект является законченной системой анимации трехмерных сцен в реальном времени. Система позволяется производить построение пространственных объектов с использованием возможностейсовременных видеокарт. Благодаря придуманной схеме взаимодействия геометрических примитивов с их материалами и свойствами, а также переносу всего вычислительного процесса на "шейдера" — вывод осуществляется с максимальным быстродействием. Система может обновлять программные "шейдера" "на лету", из-за чего изменение процесса вывода может осуществлятся без перекомпиляции всего кода. Систему можно использовать для визуализации объемных сцен и моделей, при построении произвольных анимационных проектов.

# Литература

1. David Wolff, "O+
2. OpenGL 4 Shading Language Cookbook: Build high-quality, real-time 3D graphics with OpenGL 4.6, GLSL 4.6 and C++17, 3rd Edition", Packt Publishing, 2018
3. Muhammad Mobeen Movania, "OpenGL Development Cookbook", Packt Publishing, 2013
4. John Kessenich, Graham Sellers, Dave Shreiner, "OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 4.5 with SPIR-V (9th Edition)", Addison-Wesley Professional, 2016

# Разработка проекта:

**Авторов:** 12

**Языки:** С/GLSL

(вершинные, тесселяционные, геометрические, фрагментные шейдера)

**Время разработки:** декабрь 2018-апрель 2019

**Размер кода:** 310 KiB  
 (\*.c – 200 KiB/39 файлов, \*.h – 87 KiB/29 файлов, \*.glsl – 36 KiB/22 файла)

**Строк кода:** 13000

(\*.c – 7000, \*.h - 3300)

***Разработка кода***

* Двас Павел Григорьевич, 08-2 класс  
  *подсистемы освещения, источники света, тени*
* Марков Максим Александрович, 08-2 класс  
  *подсистемы построения топологий*
* Кураленок Святослав Игоревич, 08-4 класс  
  *подсистемы вывода и низкоуровневое программировние видеоадаптера*
* Вашпанов Александр Викторович, 09-3 класс  
  *подсистемы вывода и низкоуровневое программировние видеоадаптера*
* Кононов Святослав Игоревич, 09-3 класс  
  *подсистемы построения и вывода примитивов, ввода 3-хмерного звука*
* Шайда Андрей Игоревич, 09-4 класс  
  *подсистемы материалов, текстурирования*
* Кузьмин Илья Дмитриевич, 09-5 класс  
  *подсистемы обслуживания и хранения всех ресурсво системы*
* Хлевной Евгений Александрович, 09-5 класс  
  *подсистемы построения коллекций объектов и примитивов, математические функции*
* Копейкина Софья Евгеньевна, 10-5 класс  
  *подсистемы анимации и взаимодействия с активными элементами вывода*
* Фабрика Артем Александрович, 10-5 класс  
  *подсистемы ввода с различных устройств*
* Гирвиц Александр Михайлович, 10-6 класс  
  *подсистемы обслуживания шейдеров*
* Дмитриева Василиса Валерьевна, 10-6 класс  
  *подсистемы освещения, источники света, тени*