Федеральное государственное образовательное бюджетное

учреждение высшего образования

**«Финансовый университет при Правительстве Российской Федерации»**

**(Финансовый университет)**

Колледж информатики и программирования

**ОТЧЕТ ПО УП.02.01 ЛАБОРАТОРНАЯ 1**

# **Тема: Доклад**

Студент: Кириллов К.А

Группа: 4ИСИП-619

Преподаватель: Сибирев И.В.

WebGL – это программная библиотека для JavaScript, которая позволяет создавать 3D графику, функционирующую в браузерах. Данная библиотека основана на архитектуре библиотеки OpenGL. WebGL использует язык программирования шейдеров GLSL, который имеет С-подобный синтаксис. WebGL интересен тем, что код моделируется непосредственно в браузере. Для этого WebGL использует объект canvas, который был введен в HTML5.

Работа с WebGL, и с шейдерами в частности, — это довольно трудоемкий процесс. В процессе разработки необходимо описать каждую точку, линию, грань и так далее. Чтобы все это визуализировать, нам необходимо прописать довольно объемный кусок кода. Для повышения скорости разработки, была разработана библиотека Three.js.

Three.js – это библиотека JavaScript, содержащая набор готовых классов для создания и отображения интерактивной 3D графики в WebGL.

Three.js для WebGL — это то же самое, что jQuery для JavaScript. Библиотека предлагает декларативный синтаксис, и абстрагирует от головных болей связанных с 3D в браузере.

Подробнее о Three.js.

Библиотека Three.js, как уже упоминалось, облегчает работу с WebGL. При использовании Three.js отпадает необходимость в написании шейдеров, и появляется возможность оперировать привычными понятиями.

Над библиотекой работает большое количество разработчиков. Главным идеологом и разработчиком является Ricardo Cobello, известный под творческим псевдонимом Mr.Doob.

Моделирование графики с использованием Three.js можно сравнить со съемочной площадкой, так как у нас есть возможность оперировать такими понятиями как сцена, свет, камера, объекты и их материалы.

Три, так называемых, кита Three.js включают в себя:

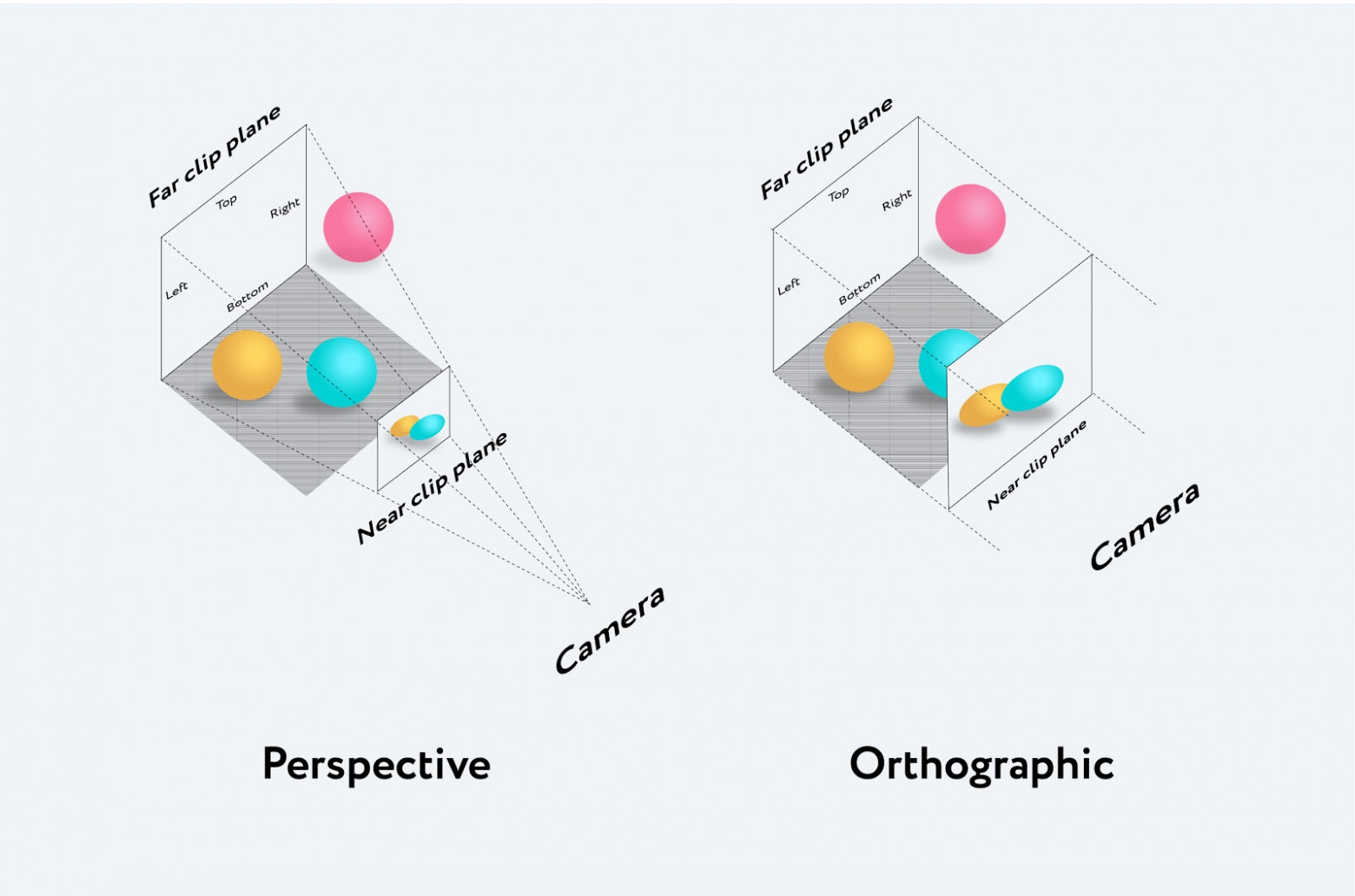
Scene - своеобразная платформа, где размещаются все объекты, которые мы создаем;

Camera - который будет направлен на сцену. Камера снимает и отображает объекты, которые расположены на сцене;

Renderer - визуализатор, который позволяет показывать сцену, снятую камерой.

В Three.js существует несколько типов камеры:

Perspective Camera,Stereo Camera,Orthographic Camera,Cube Camera. Самые распространенные из них — это Perspective Camera и Orthographic Camera.



Освещение.

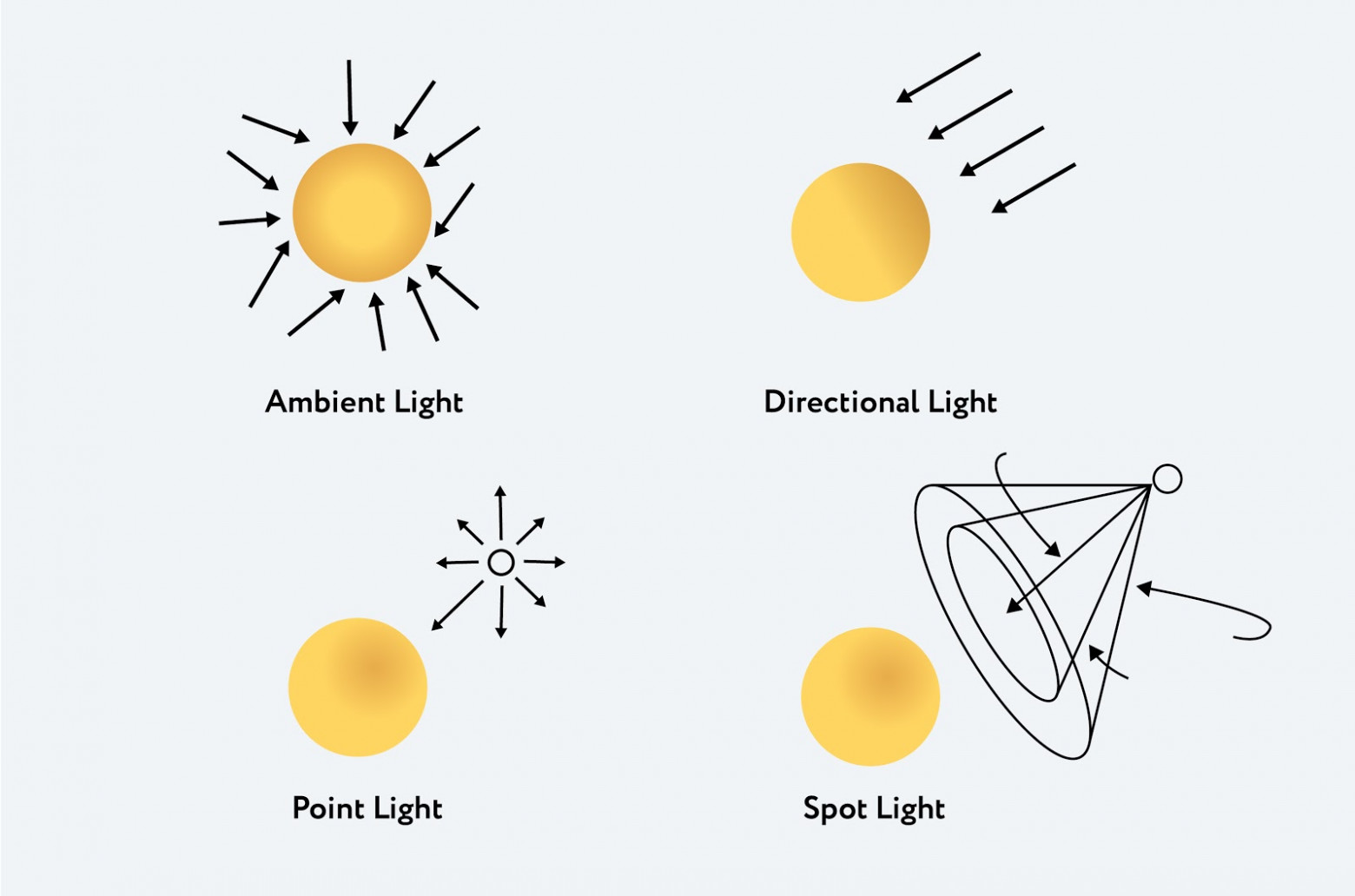
Без освещения на сцене, будет складываться впечатление, что вы находитесь в темной комнате. Помимо этого, с помощью освещения сцене можно придать большую реалистичность. Технически, каждому освещению можно задать цвет.  
  
Примеры освещения:

Ambient Light — фоновое освещение, которое используется для освещения всех объектов сцены одинаково; не может быть использован для создания теней, так как не имеет направления.

Directional Light — свет, который излучается в определенном направлении. Этот свет будет вести себя так, как если бы он был бесконечно далеко, а лучи, излучаемые из него, были параллельны; данное освещение может отбрасывать тени, так как направлено оно на конкретный объект.

Point Light — свет, который излучается из одной точки во всех направлениях. Обычный случай использования такого освещения это повторение освещения от простой лампочки (без светильника).

Spot Light — данный свет излучается из одной точки в одном направлении,вдоль конуса, расширяемого по мере удаления от источника света.



Объект, создаваемый на сцене, называется Mesh.

Mesh — это класс, представляющий объекты на основе треугольной полигональной сетки.

Этот класс принимает 2 аргумента:

Geometry — описывает форму.

Material — описывает внешний вид объектов.

Пример:

Создадим сложную фигуру.

В качестве примера возьмем материал MeshPhongMaterial, который учитывает освещенность. Поэтому, сначала нам необходимо добавить света на сцену. Ниже добавляем SpotLight с желтым оттенком и задаем ему позицию на оси координат.

var scene = new THREE.Scene();

var spotLight = new THREE.SpotLight(0xeeeece);

spotLight.position.set(1000, 1000, 1000);

scene.add(spotLight);

var spotLight2 = new THREE.SpotLight(0xffffff);

spotLight2.position.set( -200, -200, -200);

scene.add(spotLight2);

SpotLight, излучается из одной точки в одном направлении, вдоль конуса, расширяемого по мере удаления от источника света. Точечный свет помимо цвета может принимать на себя аргументы: intensity, distance, angle, penumbra, decay, а также отбрасывать тени.

Теперь определим саму фигуру:

var geometry = new THREE.TorusGeometry( 10, 3, 16, 100 );

Класс TorusGeometry предназначен для построения торусов или “валиков”. Этот класс принимает на себя следующие параметры:

* радиус, по умолчанию 1;
* диаметр трубы, по умолчанию 0.4;
* radialSegments или количество сегментов-треугольников, по умолчанию 8;
* tubularSegments или количество сегментов-граней, по умолчанию 6

Добавляем материал:

var material = new THREE.MeshPhongMaterial( {

color: 0xdaa520,

specular: 0xbcbcbc,

} );

Этот материал предназначен для блестящих поверхностей. Ему мы передаем золотистый цвет, и добавляем свойство specular, которое влияет на блеск материала и его цвет. Цвет по умолчанию — 0x111111 — темно-серый.  
  
Рендерим, и вот, что в итоге получилось:

Заключение

С помощью практически пары строчек кода мы можем создать простейшие фигуры, и без каких-либо трудностей более сложные. Three.js имеет внутри очень много фигур, материалов, типов освещения, загрузчики, математические элементы, текстуры и т. Д

Библиотека Three.js позволяет творить и создавать действительно классные вещи.

Особенности / преимущества:

Поддерживает Chrome 9+, Opera 15+, Firefox 4+, IE 11, Edge и Safari 5.1

Поддержка функций JS, таких как типизированные массивы, Blob, Promise, URL API, Fetch и многое другое