Лабораторная работа №9

«Освещение объектов в Three.js»

Оглавление

Различные виды источников света

Начнем с добавления источника света на сцену.

В Three.js доступно несколько различных видов источников света:

Объект	**Описание**
`THREE.AmbientLight`	Источник рассеянного света.
`THREE.PointLight`	Точечный источник света.
`THREE.SpotLight`	Источник направленного света.
`THREE.DirectionalLight`	Бесконечно удаленный источник света.
`THREE.AreaLight`	Распределенный источник света.

Рассмотрим эти источники света более подробно. Начнем с `THREE.AmbientLight`.

Источник рассеянного света `THREE.AmbientLight`

`THREE.AmbientLight` обычно не используется в качестве единственного источника света в сцене, поскольку он окрашивает все объекты в один цвет, независимо от формы. `THREE.AmbientLight` используется вместе с другими источниками освещения, такими как `THREE.SpotLight` или `THREE.DirectionalLight`, чтобы смягчить тени или добавить дополнительный цвет к сцене. Самый простой способ понять это – изучить пример `ambient-light.html`. В этом примере в правом верхнем углу находится меню управления, с помощью которого можно изменять интенсивность и цвет `THREE.AmbientLight`. Обратите внимание, что в этой сцене у нас также присутствует `THREE.SpotLight`, который добавляет дополнительное освещение и создает тени. С помощью меню можно выключить `THREE.SpotLight` и увидеть, какой эффект дает `THREE.AmbientLight` сам по себе.

Цвет по умолчанию, который используется в этой сцене – `#0c0c0c`. Он задается с помощью шестнадцатеричного представления цвета и представляет собой очень тусклый светло-серый цвет, который в основном используется для сглаживания жестких теней, отбрасываемых нашими объектами на плоскость земли. Вы можете изменить этот цвет, например, на более заметный желто-оранжевый цвет (`#523318`), и тогда объекты будут светиться как Солнце. Если выбранный вами цвет окажется слишком ярким, то вы получите перенасыщенное неестественное изображение.

В этом примере мы создали возможность изменения цвета `THREE.AmbientLight` с помощью меню. Для этого используется та же библиотека `dat.GUI`, что и в предыдущих лабораторных работах. Единственное изменение заключается в том, что вместо функции `gui.add(…)` для выбора цвета используется функция `gui.addColor(…)`. Она создает параметр в меню управления, с помощью которого можно напрямую изменять цвет переданной переменной. В коде вы можете заметить, что здесь добавляется обработчик события `onChange` для элемента: `dat.GUI: gui.addColor(…).onChange(function(e){…})`. С помощью этой функции мы говорим `dat.GUI` вызывать переданную функцию каждый раз при изменении цвета. В данном случае мы присваиваем цвету `THREE.AmbientLight` новое значение.

9.1 Задание для самостоятельной работы

Добавьте в программу вашего домашнего задания рассеянное освещение.

Точечный источник света `THREE.PointLight`

Чтобы поэкспериментировать с точечным источником света `THREE.PointLight`, можно использовать пример `point-light.html`, где точечный источник света перемещается по сцене. Чтобы было понятно, где находится точечный источник света, мы перемещаем маленькую

оранжевую сферу по тому же пути. По мере того, как этот свет перемещается, вы увидите, что красный куб и синяя сфера освещаются этим светом с разных сторон.

`THREE.PointLight` имеет несколько параметров конфигурации:

Свойство	Описание
`color`	Цвет света.
`distance`	Максимальное расстояние, на которое распространяется свет от источника. Значение по умолчанию равно `0`, что означает, что затухание света отсутствует.
`intensity`	Интенсивность света (кд).
`position`	Положение источника света на сцене.
`visible`	`true` (по умолчанию) – свет включен, `false` – свет выключен.

`THREE.PointLight` можно создать с помощью следующей последовательности команд:

```
const pointColor = "#ccffcc";

const pointLight = new THREE.PointLight(pointColor, 100);

pointLight.position.set(10,10,10);

scene.add(pointLight);
```

Здесь мы создаем источник света определенного цвета, устанавливаем его свойство `position` и добавляем его на сцену.

Рассмотрим теперь свойство интенсивности. С помощью этого свойства можно установить, насколько ярко светит свет. Значение интенсивности задается в канделах.

Свойство `distance` объекта `THREE.PointLight` определяет, как далеко распространяется свет от источника, прежде чем его интенсивность станет равной `0`. Это свойство можно установить следующим образом:

```
pointLight.distance = 14;
```

В этом случае, интенсивность света будет равномерно уменьшаться до `0` на расстоянии `14`. Значение по умолчанию для свойства `distance` равно `0`, что означает, что свет не будет затухать на расстоянии.

Отбрасывание тени

Рендеринг теней требует больших вычислительных мощностей GPU, поэтому их расчет по умолчанию отключен в Three.js. Однако включить их расчет несложно. Для этого мы должны выполнить несколько команд.

Первая команда, которую нам необходимо сделать, это сообщить рендеру, что нам нужны тени. Для этого, нужно присвоить свойству `shadowMapEnabled` значение `true`. В примере `point-light.html` нужно раскомментировать строку:

```
// renderer.shadowMapEnabled = true;
```

Далее нужно явно определить, какие объекты отбрасывают тени и на какие объекты тени накладываются. В примере `point-light.html` мы хотим, чтобы сфера и куб отбрасывали тени на плоскость земли. Мы делаем это, устанавливая свойства `castShadow = true` для сферы и куба и `receiveShadow = true` для плоскости.

И последнее, нам нужно определить, свет от каких источников будет вызывать тени. Не все источники света могут давать тени. `THREE.AmbientLight` не дает никаких теней. Источники `THREE.PointLight`, `THREE.SpotLight` и `THREE.DirectionalLight` могут давать тени.

Чтобы включить отбрасывание тени от источника, нужно присвоить свойству `castShadow` значение `true`, например:

```
pointLight.castShadow = true;
```

Когда включено отбрасывание тени от источника света, то можно управлять тем, как эта тень будет отображаться. Это делается с помощью свойства `shadow` источника:

Свойство	**Описание**
`shadow.camera.far`	Определяет, на каком максимальном расстоянии от источника света должны создаваться тени. Значение по умолчанию – `5000`.
`shadow.camera.fov`	Определяет, насколько велико поле обзора, используемое для создания теней. Значение по умолчанию – `50`.
`shadow.camera.near`	Определяет, на каком минимальном расстоянии от света должны создаваться тени. Значение по умолчанию – `50`.
`shadow.camera.visible`	Если установлено значение `true`, то можно увидеть, как и где этот источник света отбрасывает тень. Значение по умолчанию `false`.
`shadow.mapSize.width` и `shadow.mapSize.height`	Определяет, сколько пикселей используется для создания тени. Увеличьте это значение, если тень имеет неровные края или не выглядит гладкой. Это значение нельзя изменить после рендеринга сцены. Значение по умолчанию – 512 для обоих свойств.

9.2 Задание для самостоятельной работы

Добавьте в программу вашего домашнего задания точечный источник света.

Источник направленного света `THREE.SpotLight`

В следующей таблице перечислены свойства `THREE.SpotLight`:

Свойство	**Описание**
`angle`	Определяет угловую ширину конуса света θ_l. Измеряется в радианах и по умолчанию равна `Math.PI/3` (см. рис. 1).
`penumbra`	Степень углового затухания интенсивности света a_l.
`color`	Цвет света.
`distance`	Максимальное расстояние, на которое распространяется свет от источника. Значение по умолчанию равно 0, что означает, что затухание света отсутствует (см. рис. 1).
`intensity`	Интенсивность света. Значение по умолчанию равно 1.
`onlyShadow`	Если для этого свойства установлено значение `true`, источник света будет только отбрасывать тень и не будет добавлять свет в сцену.
`position`	Положение источника света на сцене.
`target`	При использовании `THREE.SpotLight` важно направление, в котором он светит. С помощью этого свойства можно указать источнику светить на конкретный объект или позицию в сцене.
`visible`	`true` (по умолчанию) – свет включен, `false` – свет выключен.

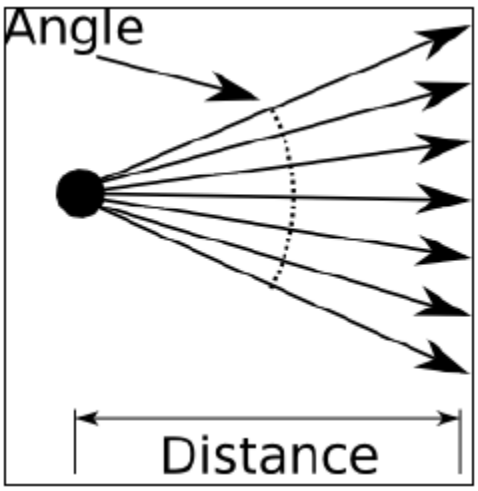

Рис. 1. Область освещения, которую формирует источник `THREE.SpotLight`

Для создания направленного источника света нужно указать его цвет, установить нужные свойства и добавить его в сцену следующим образом:

```
const spotLight = new THREE.SpotLight("#ffffff");
spotLight.position.set(-40, 60, -10);
spotLight.castShadow = true;
spotLight.shadow.camera.near = 1;
spotLight.shadow.camera.far = 100;
spotLight.target = plane;
spotLight.distance = 0;
spotLight.angle = 0.4;
spotLight.shadow.camera.fov = 120;
scene.add(spotLight);
```

Здесь мы создали экземпляр `THREE.SpotLight` и установили различные свойства для настройки света. Мы также явно установили для свойства `castShadow` значение `true`, потому что нам нужны тени. Свойство `target` наводит источник света `THREE.SpotLight` на середину плоскости. Откройте пример (`spot-light.html`) и посмотрите на полученную сцену.

Можно направить источник `THREE.SpotLight` на другой объект на сцене. Например, мы можем направить его на центр синей сферы, тогда свет будет все время фокусироваться на ней, даже несмотря на то, что сфера перемещается по сцене. Если нужно направить свет не на конкретный объект, а на произвольную точку пространства, то это можно сделать следующим образом:

```
const target = new THREE.Object3D();
target.position.set(5, 0, 0);
spotlight.target = target;
```

При отладке проблем с тенями от источника света можно использовать объект `THREE.CameraHelper`. Для этого достаточно добавить следующие строки:

```
const debugCamera = new THREE.CameraHelper(spotLight.shadow.camera);
```

```
scene.add(debugCamera);
```

Установив флажок `shadowDebug` в меню, можно увидеть область, которая используется для определения теней от этого источника света.

Для подбора значений параметров источника `THREE.SpotLight`, влияющих на его форму и направление, Three.js также предоставляет объект `THREE.SpotLightHelper`. Для добавления этого помощника на сцену, можно использовать следующий фрагмент кода:

```
const helper = new THREE.SpotLightHelper(spotLight);

scene.add(helper);

function render() {

  ...

  helper.update();

  ...}
```

9.3 Задание для самостоятельной работы

Добавьте в программу вашего домашнего задания направленный источник света.

Определение бесконечно удаленного источника света THREE.DirectionalLight

`THREE.DirectionalLight` определяет источник света, который расположен очень далеко по отношению к размерам сцены. Все световые лучи, которые он испускает, параллельны друг другу. Примером такого источника света является Солнце. Солнце находится так далеко, что световые лучи, которые мы получаем на Земле, (почти) параллельны друг другу. Основное различие между `THREE.DirectionalLight` и `THREE.SpotLight` заключается в том, что этот свет не будет угасать по мере удаления от источника `THREE.DirectionalLight`, как это происходит с `THREE.SpotLight` (если для него настроено угасание). Вся область, освещенная `THREE.DirectionalLight`, получает одинаковую интенсивность света.

Пример работы с этим источником света приведен в файле `directional-light.html`.

`THREE.DirectionalLight` имеет множество свойств, таких же, как у `THREE.SpotLight`, главными из которых являются свойства: `position`, `target`, `intensity`.

Поскольку для `THREE.DirectionalLight` все лучи параллельны друг другу, у нас нет светового конуса; вместо этого у нас есть кубическая область, которую можно увидеть, если установить флажок отладки). Все, что попадает в этот куб, может отбрасывать и принимать тени от источника света. Как и для `THREE.SpotLight`, чем уже будет определена эта область вокруг объектов, тем лучше будут выглядеть тени. Определим этот куб, используя следующие свойства:

```
directionalLight.castShadow = true;

directionalLight.shadow.camera.near = 2;

directionalLight.shadow.camera.far = 80;

directionalLight.shadow.camera.left = -30;

directionalLight.shadow.camera.right = 30;

directionalLight.shadow.camera.top = 30;

directionalLight.shadow.camera.bottom = -30;
```

9.4 Задание для самостоятельной работы

Добавьте в программу вашего домашнего задания бесконечно удаленный источник света.

Распределенный источник света THREE.AreaLight

С помощью THREE.AreaLight мы можем определить распределенный источник света, представляющий собой прямоугольную область. В файле `area-light.html` приведен пример работы с этим источником света.

Мы определили три объекта `THREE.AreaLight`, каждый из которых имеет свой собственный цвет. Для добавления нового распределенного источника света на сцену используется следующий код:

```
const areaLight1 = new THREE.RectAreaLight(0xff0000, 500, 4, 10);
areaLight1.position.set(-10, 10, -35);
scene.add(areaLight1);
```

Здесь цвет источника имеет значение `0xff0000`, интенсивность – `500` кд/м^2, а ширина и высота источника света имеют значения `4` и `10` соответственно. Также как и для других источников света, свойство `position` определяет положение источника света на сцене. Источник создается в вертикальной плоскости. Сам источник света на сцене не виден, видно только взаимодействие света от него с объектами на сцене. Чтобы визуализировать источник света следует добавить `THREE.PlaneGeometry` или `THREE.BoxGeometry` в тех же местах, где расположены источники света. Например, для визуализации `areaLight1` мы добавили следующий код:

```
const planeGeometry1 = new THREE.BoxGeometry(4, 10, 0);
const planeGeometry1Mat = new THREE.MeshBasicMaterial({color:
                                                        0xff0000});
const plane1 = new THREE.Mesh(planeGeometry1, planeGeometry1Mat);
plane1.position.copy(areaLight1.position);
scene.add(plane1);
```

9.5 Задание для самостоятельной работы

Добавьте в программу вашего домашнего задания распределенный источник света.

Настройка материалов в Three.js

Ранее мы уже начали использовать различные материалы. Материал совместно с объектом `THREE.Geometry` формирует объект `THREE.Mesh`. Материал влияет на то, как будут выглядеть поверхности объекта и имитировать физический материал, из которого изготовлен объект. Рассмотрим следующие материалы:

Свойство	**Описание**
`MeshBasicMaterial`	Одноцветный материал.
`MeshLambertMaterial`	Матовый материал.
`MeshPhongMaterial`	Блестящий материал.

Для настройки свойств этих материалов, можно использовать два способа:

1. Можно передать аргументы сразу в конструктор объекта материала, например:

```
const material = new THREE.MeshBasicMaterial(
```

```
{
      color: 0xff0000, name: 'material-1', opacity: 0.5,
      transparency: true, ...
});
```

2. Можно создать экземпляр класса и установить свойства по отдельности, например:

```
const material = new THREE.MeshBasicMaterial();
material.color = new THREE.Color(0xff0000);
material.name = 'material-1';
material.opacity = 0.5;
material.transparency = true;
```

Обычно лучше использовать первый способ, если мы знаем значения всех свойств при создании материала. Аргументы, используемые в обоих случаях, используют один и тот же формат. Единственным исключением из этого правила является свойство цвета. В первом случае мы можем просто передать шестнадцатеричное значение, и Three.js сам создаст объект `THREE.Color`. Во втором случае мы должны явно создать объект `THREE.Color`.

Изменение материалов всех объектов сцены

С помощью свойства `overrideMaterial` объекта сцены можно заставить все объекты сцены использовать один и тот же материал и игнорировать значения материалов объектов, например:

```
scene.overrideMaterial = new THREE.MeshLambertMaterial({color:
                                                        0xffffff});
```

В примере `forced-materials.html` с помощью элементов управления можно добавить куб на сцену, удалить последний добавленный на сцену куб и отобразить все текущие объекты, содержащиеся на сцене, в консоли браузера. Последняя строка элементов управления показывает текущее количество объектов на сцене.

При нажатии кнопки `addCube`, создается новый объект `THREE.BoxGeometry`, ширина, высота и глубина которого устанавливаются как случайное значение от 1 до 3. Помимо случайного размера, куб также получает случайное положение.

Все кубы визуализируются с использованием одного и того же материала и одного цвета. Это достигается благодаря установке свойства `scene.overrideMaterial`.

Следующая функция, которую мы можем вызвать из графического интерфейса управления — это `removeCube`. Как следует из названия, нажатие на кнопку `removeCube` удаляет со сцены последний добавленный куб.

Поскольку сцена (`THREE.Scene`) хранит свои дочерние элементы в виде списка (новые добавляются в конец), мы можем использовать свойство `children`, содержащее массив всех объектов сцены, чтобы получить последний объект, который был добавлен. Нам также нужно проверить, является ли этот объект объектом `THREE.Mesh`, чтобы избежать удаления камеры и источников света. После того, как мы удалили объект, мы обновляем свойство графического интерфейса `numberOfObjects`, которое содержит количество объектов на сцене.

Последняя кнопка в нашем графическом интерфейсе называется `outputObjects`. При нажатии на эту кнопку в консоль веб-браузера выводится информация обо всех объектах, которые в данный момент находятся на сцене.

Для обновления поворота каждого из кубов используется функция `render()`.

При рендеринге сцены используется метод `THREE.Scene.traverse()`. Мы можем передать функцию методу `traverse()`, которая будет вызываться для каждого дочернего элемента сцены (обратите внимание, что здесь явно исключается плоскость основания).

Основные свойства материалов

Three.js предоставляет базовый класс материала `THREE.Material`, в котором перечислены все общие свойства. Рассмотрим некоторые из них:

Свойство	**Описание**
`opacity`	Определяет, насколько прозрачен материал. Используется вместе со свойством `transparent`. Диапазон изменения этого свойства — от 0 до 1.
`transparent`	Если установлено значение `true`, Three.js будет отображать этот объект с прозрачностью, заданной свойством `opacity`.
`visible`	Определяет, будет ли материал видимым. Если параметр имеет значение `false`, материал будет не виден.
`side`	С помощью этого свойства можно определить, к какой стороне геометрии применяется материал. По умолчанию используется значение `THREE.FrontSide`, которое применяет материал к лицевой (наружней) стороне объекта. Для этого параметра также можно установить значение `THREE.BackSide`, которое применяет его к задней (внутренней) стороне, или `THREE.DoubleSide`, который применяет материал к обеим сторонам. Можно проверить, как работает это свойство, выбрав плоскую сетку как в программе `sides.html`. Поскольку обычно материал наносится только на лицевую сторону объекта, вращающаяся плоскость будет невидима половину времени (когда мы видим обратную сторону). Если вы установите для свойства `side` значение `THREE.DoubleSide`, плоскость будет видна все время, так как материал применяется к обеим сторонам геометрии. В этом случае рендерер выполняет больше работы, так что это может повлиять на производительность отрисовки сцены.
`needsUpdate`	Большинство свойств материала изменяются во время выполнения. Однако некоторые из них (например, `side`) не изменяются. Если вы хотите, чтобы изменения, внесенные вами в материал, приводили к обновлению, нужно установить для этого свойства значение `true`.

THREE.MeshBasicMaterial

Базовый материал (`THREE.MeshBasicMaterial`) никак не реагирует на источники света сцены. Он просто отображает материал заданного цвета. У этого материала есть следующие свойства:

Свойство	**Описание**
`color`	Цвет материала.
`wireframe`	Показать каркасный вид поверхности. Удобно использовать для отладки.

Следующая команда создает новый базовый материал фиолетового цвета (0x7777ff):

```
const meshMaterial = new THREE.MeshBasicMaterial({color: 0x7777ff});
```

В программе `basic-mesh-material.html` показан вращающийся куб, грани которого состоят из базового материала. Программа позволяет изменять свойства `THREE.MeshBasicMaterial` и основные свойства базового объекта `THREE.Material`.

9.6 Задание для самостоятельной работы

В программе вашего домашнего задания создайте поверхность фигуры из базового материала.

THREE.MeshLambertMaterial

Этот материал можно использовать для создания матовых, неблестящих поверхностей. Он зависит от источников освещения сцены.

Материал настраивается с помощью следующих двух свойств:

Name	Description
`color`	Коэффициент диффузного отражения k_d.
`emissive`	Интенсивность света, который излучает сам материал I_{surf}. По умолчанию эта интенсивность равна нулю.

Этот материал можно создать, например, следующим образом:

```
const meshMaterial = new THREE.MeshLambertMaterial({color: 0x7777ff});
```

В файле `mesh-lambert-material.html` приведен пример программы для работы с этим материалом.

`THREE.LambertMaterial` также работает и для каркасных изображений, поэтому с его помощью можно визуализировать каркасное изображение, которое реагирует на освещение сцены.

9.7 Задание для самостоятельной работы

В программе вашего домашнего задания создайте матовую поверхность фигуры.

THREE.MeshPhongMaterial

`THREE.MeshPhongMaterial` позволяет создать блестящий материал.

Свойства этого материала показаны в следующей таблице:

Name	Description
`color`	Коэффициент диффузного отражения k_d.
`emissive`	Интенсивность света, который излучает сам материал I_{surf}. По умолчанию эта интенсивность равна нулю.
`specular`	Коэффициент зеркального отражения k_s.
`shininess`	Показатель зеркального отражения m. Значение по умолчанию равно 30.

Инициализация объекта `THREE.MeshPhongMaterial` выполняется так же, как для всех других материалов:

```
const meshMaterial = new THREE.MeshPhongMaterial({color: 0x7777ff});
```

В файле `mesh-phong-material.html` приведен пример программы для работы с этим материалом.

9.8 Задание для самостоятельной работы

В программе вашего домашнего задания создайте блестящую поверхность фигуры.

Добавление эффекта тумана

Эффект тумана позволяет скрывать объекты по мере их удаления от камеры.

Чтобы включить туман в Three.js требуется добавить следующую строку кода:

```
scene.fog=new THREE.Fog(0xffffff, 0.015, 100);
```

Она определяет туман белого цвета (`0xffffff`). Следущие два свойства можно использовать для настройки появления тумана. Значение 0.015 задает ближнюю границу d_{min}, а значение 100 задает дальнюю границу d_{max}. С помощью этих свойств можно определить, где начинается туман и как быстро он становится гуще. Объект `THREE.Fog` задает линейную функцию затухания. Существует также туман с экспоненциальной функцией затухания для сцены:

```
scene.fog=new THREE.FogExp2(0xffffff, 0.01);
```

Параметрами этого тумана являются его цвет (`0xffffff`) и плотность (`0.01`). Можно поэкспериментировать с этими свойствами, чтобы получить желаемый эффект.

9.9 Задание для самостоятельной работы

Вариант 1. В программу вашего домашнего задания добавьте линейную функцию тумана. Задайте необходимые параметры тумана, чтобы его эффект был виден на визуализируемой сцене.

Вариант 2. В программу вашего домашнего задания добавьте экспоненциальную функцию тумана. Задайте необходимые параметры тумана, чтобы его эффект был виден на визуализируемой сцене.