УДК 004.92
DOI 10.51691/2541-8327_2023_6_20

ТРЕХМЕРНОЕ МОДЕЛИРОВАНИЕ В БРАУЗЕРЕ С ПОМОЩЬЮ THREE.JS

Вильданов А.Н.
к.ф.-м.н., доцент
ФГБОУ ВО «Уфимский университет науки и технологий», Нефтекамский филиал
Нефтекамск, Россия
Исмагилов А.И.
студент
ФГБОУ ВО «Уфимский университет науки и технологий», Нефтекамский филиал
Нефтекамск, Россия

Аннотация: в данной работе описываются примеры работы с Three.js. Three.js – это библиотека JavaScript, которая облегчает создание и отображение 3D-графики веб-браузере. Она предоставляет мощные инструменты и API для создания интерактивных и визуально привлекательных веб-приложений, игр, визуализаций и других проектов, основанных на 3D-графике. Работа предоставляет читателям представление о библиотеке, ее возможностях и способах использования, чтобы они могли успешно использовать Three.js на своих веб-страницах и эффективно отображать трехмерные фигуры. Результаты, изложенные в статье, могут быть полезны как для преподавателей, так и для учащихся, так как Three.js может быть применена в образовательных проектах для трехмерной визуализации и наглядной демонстрации сложных математических концепций, физических явлений или научных данных.

Ключевые слова: JavaScript, WebGL, Three.js, 3D-моделирование, компьютерная графика.

3D MODELING IN THE BROWSER WITH THREE.JS

Vildanov A.N.
candidate of Physical and Mathematical Sciences,
Ufa University of Science and Technology, Neftekamsk branch
Neftekamsk, Russia
Ismagilov A.I.
student,
Ufa University of Science and Technology, Neftekamsk branch
Neftekamsk, Russia

Abstract: this paper describes examples of working with Three.js. Three.js is a JavaScript library that makes it easy to create and display 3D graphics in a web browser. It provides powerful tools and APIs for creating interactive and visually appealing web applications, games, visualizations, and other 3D graphics-based projects. The work provides readers with an understanding of the library, its capabilities, and how to use it so that they can successfully use Three.js in their web pages and display 3D shapes efficiently. The results presented in the article can be useful for both teachers and students, since Three.js can be used in educational projects for three-dimensional visualization and visual demonstration of complex mathematical concepts, physical phenomena or scientific data.

Keywords: JavaScript, WebGL, Three.js, 3D modeling, computer graphics.

Введение. В 2011 году, с целью предоставить возможность разработчикам создавать и отображать 3D-графику прямо в веб-браузерах, используя ресурсы графического процессора (GPU) компьютера, был разработан WebGL. WebGL позволяет создавать сложные и производительные 3D-приложения, такие как

визуализация данных, игры, симуляции и виртуальная реальность, непосредственно в браузере [1].

Для работы с WebGL требуется поддержка со стороны браузера и графического адаптера компьютера или мобильного устройства. Большинство современных браузеров, таких как Google Chrome, Mozilla Firefox, Safari и Opera, поддерживают WebGL.

Three.js – это библиотека JavaScript, которая предоставляет разработчикам инструменты для создания и отображения 3D-графики веб-браузере [2].

Какие задачи и проблемы Three.js помогает решить? Программирование 3D-графики требует знания специализированных графических API, таких как WebGL. Three.js абстрагирует сложности работы с этими API и предоставляет простой и понятный API на основе JavaScript, что облегчает разработку 3D-графики для веб-браузера.

Three.js также решает проблему кросс-браузерной совместимости, предоставляя унифицированный и согласованный подход к разработке 3D-графики, который работает во всех современных браузерах.

Библиотека Three.js предоставляет удобные инструменты для управления сложными 3D-сценами, объектами и анимациями. Она позволяет разработчикам создавать и контролировать движение, поворот, масштабирование и другие визуальные эффекты, делая 3D-графику более интерактивной и привлекательной.

Наконец, Three.js предоставляет возможности для взаимодействия с пользователем через мышь, клавиатуру и другие устройства ввода. Разработчики могут создавать интерактивные 3D-приложения и игры, которые реагируют на действия пользователей.

Основные возможности и функциональность. Three.js предоставляет множество ключевых возможностей для работы с 3D-графикой веб-браузера. Вот некоторые из них:

1. Создание и управление сценами: Three.js позволяет создавать и управлять трехмерными сценами, которые включают в себя объекты, источники света и камеры. Разработчики могут настраивать положение, вращение, масштабирование и другие параметры сцены и ее компонентов.

2. Создание и манипуляция 3D-объектами: Three.js предоставляет возможность создавать и манипулировать различными типами 3D-объектов, такими как геометрические формы (кубы, сферы, цилиндры и т.д.), и многое другое. Разработчики могут настраивать внешний вид и свойства объектов, добавлять текстуры, материалы и трансформации.

3. Материалы и текстуры: Three.js предоставляет богатый набор материалов и текстур для применения к 3D-объектам. Разработчики могут выбирать и настраивать различные типы материалов, а также можно применять текстуры для добавления деталей и реалистичности к объектам.

4. Освещение и тени: Three.js поддерживает различные источники света, такие как направленные, точечные и прожекторные источники. Разработчики могут настраивать параметры освещения и создавать эффекты теней, отражений и преломлений.

5. Анимация: Three.js обеспечивает возможность создания анимаций 3D-объектов. Доступны возможности для создания сложных анимаций, скелетной анимации и морфинга.

6. Взаимодействие с пользователем: Three.js позволяет разработчикам реагировать на действия пользователей и создавать интерактивные 3D-приложения. Это включает обработку событий мыши, клавиатуры и сенсорных устройств, реагирование на клики и перемещения.

7. Импорт готовых моделей из внешних файлов (например, форматы OBJ, FBX), разработанных в других программах для трехмерного моделирования, таких как, Blender, Maya, 3ds Max и т.д.

Эти и многие другие из ключевых возможностей Three.js делают ее мощным инструментом для создания 3D-графики в веб-браузере.

Подключение. Будет загружать модуль Three.js с помощью карты импортов (Import maps) – механизма, позволяющего получить контроль над поведением JavaScript-импортов:

```
<script type="importmap">
     {
          "imports": {
               "three": "https://threejs.org/build/three.module.js"
          }
     }
</script>
```

Далее импортируем все объекты из модуля three и присвоим их переменной THREE. Модуль three содержит основные классы и функции для работы с Three.js.

```
import * as THREE from 'three';
```

Также импортируем классы OrbitControls и TeapotGeometry для дальнейшего использования в создании трехмерных объектов и управления камерой в Three.js. OrbitControls предоставляет интерактивное управление камерой в Three.js, позволяя вращать, масштабировать и перемещать камеру вокруг целевой точки. TeapotGeometry представляет геометрию для трехмерного чайника:

```
import { OrbitControls } from
   'https://threejs.org/examples/jsm/controls/OrbitControls.js'
import { TeapotGeometry } from
   'https://threejs.org/examples/jsm/geometries/TeapotGeometry.js';
```

Создание сцены. Инициализируем основные компоненты для создания и отображения трехмерной сцены с использованием Three.js, такие, как контейнер, камера, источники света, рендерер, элементы управления и сама сцена.

Сначала создается новый блок , который будет служить контейнером для отображения сцены. Этот контейнер добавляется в тело документа:

```
var container = document.createElement( 'div' );
document.body.appendChild( container );
```

Далее создается «перспективная» камера с определенными параметрами, такими, как угол зрения в 45°, соотношение сторон (window.innerWidth / window.innerHeight), ближняя и дальняя плоскости отсечения (1 и 5000 соответственно). Затем устанавливается позиция камеры в трехмерном пространстве:

```
camera = new THREE.PerspectiveCamera(
          45, window.innerWidth / window.innerHeight, 1, 5000 );
camera.position.set( 300, 700, 1200 );
```

Создадим фоновое освещение AmbientLight нейтрального серого цвета и с интенсивностью 0.2 [4]. Также добавим более яркий, направленный белый свет DirectionalLight и интенсивностью 1. При создании света укажем позицию источника света в трехмерном пространстве:

```
ambientLight = new THREE.AmbientLight( 0x333333 ); // 0.2
light = new THREE.DirectionalLight( 0xFFFFFF, 1.0 );
light.position.set( 1, 1, 1 );
```

Далее создадим отрисовщик WebGL-рендерер с опцией сглаживания (antialias: true). Устанавливается соотношение пикселей устройства и размеры

рендерера, чтобы они соответствовали размерам окна. Рендерер добавляется в контейнер, созданный на первом шаге:

```
renderer = new THREE.WebGLRenderer( { antialias: true } );
renderer.setPixelRatio( window.devicePixelRatio );
renderer.setSize( window.innerWidth, window.innerHeight );
container.appendChild( renderer.domElement );
```

Далее добавляется слушатель события изменения размера окна (resize). При возникновении события будет вызвана функция onWindowResize, которая будет обновлять соотношение и размеры рендерера:

```
window.addEventListener( 'resize', onWindowResize, false );
```

Для более удобного и наглядного просмотра сцены добавим на сцену элемент управления OrbitControls, который позволяет пользователю вращать и приближать сцену с помощью мыши или жестов. Устанавливаются параметры данного элемента управления, такие, как скорость вращения (rotateSpeed), возможность приближения (enableZoom), скорость приближения (zoomSpeed), минимальное и максимальное расстояния для приближения (minDistance и maxDistance соответственно), и включение (затухающего) движения камеры по инерции при просмотре (enableDamping):

```
controls = new OrbitControls( camera, renderer.domElement );
controls.addEventListener( 'change', render );
controls.rotateSpeed = 0.5;
controls.enableZoom = true;
controls.zoomSpeed = 0.5;
controls.minDistance = 500;
controls.maxDistance = 2500;
controls.enableDamping = true;
```

Наконец, создается новая сцена Scene и устанавливается цвет фона сцены. Затем окружающий свет и направленный свет добавляются в созданную сцену:

```
scene = new THREE.Scene();
scene.background = new THREE.Color( 0xD3D3D3 );
scene.add( ambientLight );
scene.add( light );
```

Добавление трехмерных объектов на сцену. Для создания трехмерного примитива в Three.js задаются геометрия, материал и объект Mesh. Рассмотрим пример создания трехмерного кубоида (прямоугольного параллелепипеда).

Сначала создадим геометрию кубоида:

```
var geometry = new THREE.BoxGeometry( 200, 200, 150 );
```

Здесь указываются значения width, height и depth (ширина, высота и глубина) – это параметры, определяющие размеры кубоида.

Далее создадим красно-оранжевый материал для кубоида:

```
var material = new THREE.MeshPhongMaterial( { color: 0xFF4500 } );
```

Далее, создается меш (объект) кубоида, используя геометрию и материал:

```
var Cube = new THREE.Mesh( geometry, material );
```

Меш объединяет геометрию и материал и представляет собой конкретный объект, который может быть добавлен на сцену. Установим позицию кубоида в трехмерном пространстве:

```
Cube.position.set( -210, 100, -150 );
```

Здесь указываются x, y и z - это координаты позиции кубоида. Повернем кубоид на 140° (если смотреть сверху, со стороны оси Oy):

```
Cube.rotation.y = Math.PI / 140;
```

Следующий шаг добавляет созданный куб на сцену, чтобы он был отображен:

```
scene.add( Cube );
```

После выполнения этих шагов, трехмерный куб будет добавлен на сцену в Three.js и будет отображаться соответствующим образом в соответствии с выбранными параметрами геометрии и материала (рисунок 1):

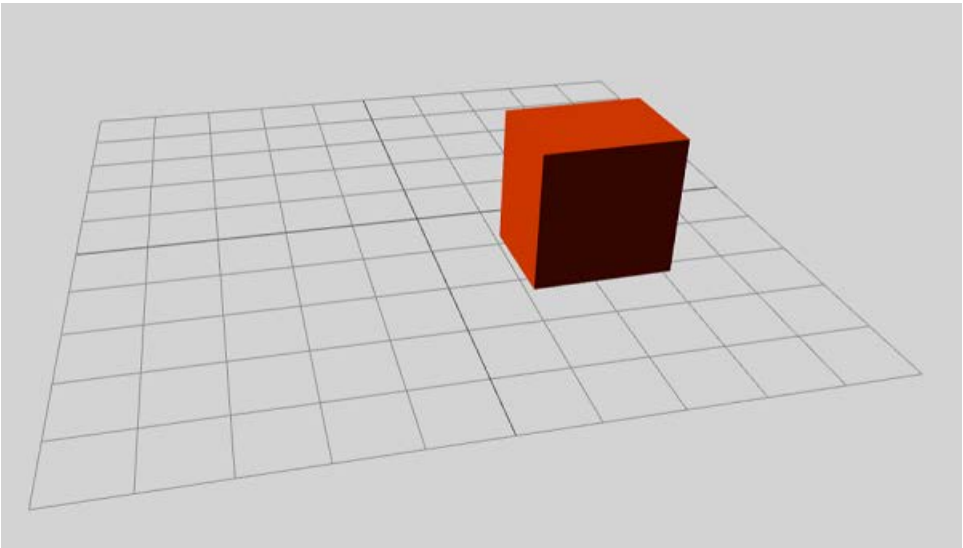

Рисунок 1 – Кубоид

Здесь также добавлена сетка, которая отображает горизонтальные и вертикальные линии, образующие сетку, и помогает визуализировать и ориентироваться в трехмерном пространстве:

```
const helper = new THREE.GridHelper( 1000, 10 );
scene.add( helper );
```

Данный код создает сетку GridHelper на сцене в Three.js и добавляет ее в сцену. В конструкторе GridHelper указаны два параметра:

– размер сетки (1000) - это длина и ширина сетки в единицах трехмерного пространства;

– шаг (10) - это расстояние между линиями сетки.

Далее, по аналогичной схеме, создадим и добавим на сцену сферу с помощью класса SphereGeometry (рисунок 2):

```
var geometry = new THREE.SphereGeometry(100, 50, 50);
var material = new THREE.MeshPhongMaterial( { color: 0xc41e3a } );
var Sphere1 = new THREE.Mesh( geometry, material );
Sphere1.position.set( 350, 50, -100 );
scene.add( Sphere1 );
```

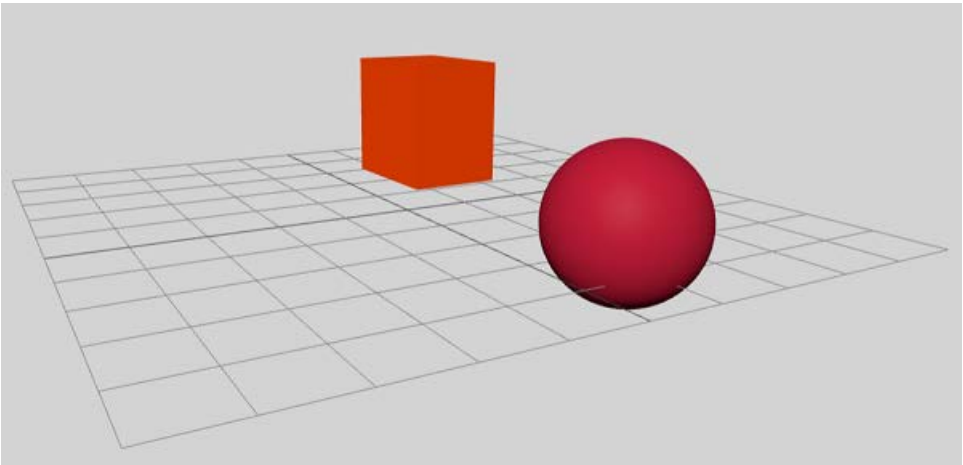

Рисунок 2 – Сфера

Такие трехмерные фигуры, как пирамиды, призмы, цилиндра и конуса создаются с помощью одной и той же команды CylinderGeometry [3]. Создадим, например, конус (рисунок 3):

```
var radiusTop = 0; var radiusBottom = 128;
var heigth = 200; var segments = 16;
var geometry = new THREE.CylinderGeometry(
                      radiusTop, radiusBottom, heigth, segments );
var material = new THREE.MeshPhongMaterial( { color: 0x177245 } );
var prizm = new THREE.Mesh( geometry, material );
prizm.position.set( 50, 100, -130 );
scene.add( prizm );
```

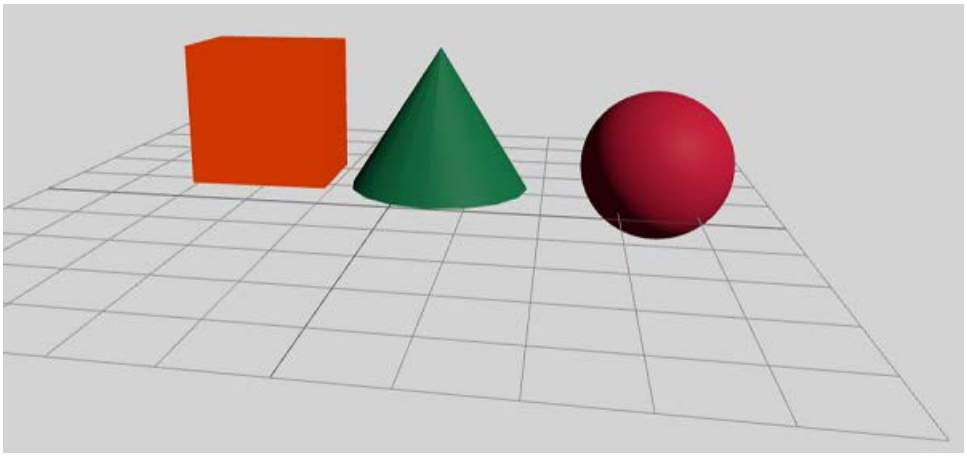

Рисунок 3 – Конус

Указываются следующие параметры конуса:

– radiusTop - радиус верхней окружности цилиндра (в данном случае равен 0).

– radiusBottom - радиус нижней окружности цилиндра (в данном случае равен 128).

– height – высота цилиндра (в данном случае равна 200).

– segments – количество сегментов, на которые разделены окружности цилиндра (в данном случае равно 16).

Чем больше сегментов, тем больше полученная фигура похожа на конус. Если сегментов всего три, мы получим треугольную пирамиду.

Добавим также фигуру «Чайник», который является популярным объектом в трехмерном моделировании, и часто используется для различных целей (рисунок 4):

```
var geometry = new TeapotGeometry( 100 );
var material = new THREE.MeshPhongMaterial( { color: 0xffccff } );
var teapot = new THREE.Mesh( geometry, material );
teapot.position.set( 650, 50, 200 );
scene.add( teapot );
```

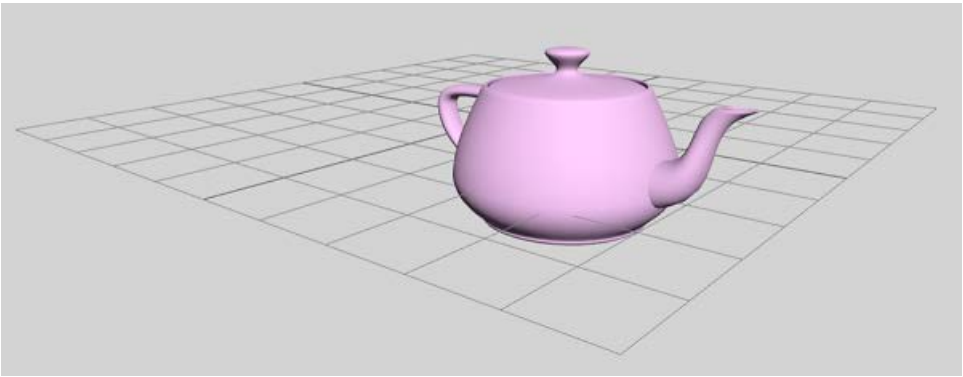

Рисунок 4 – Чайник

Создадим аналогично фигуры пирамиды, тора и цилиндров (рисунок 5):

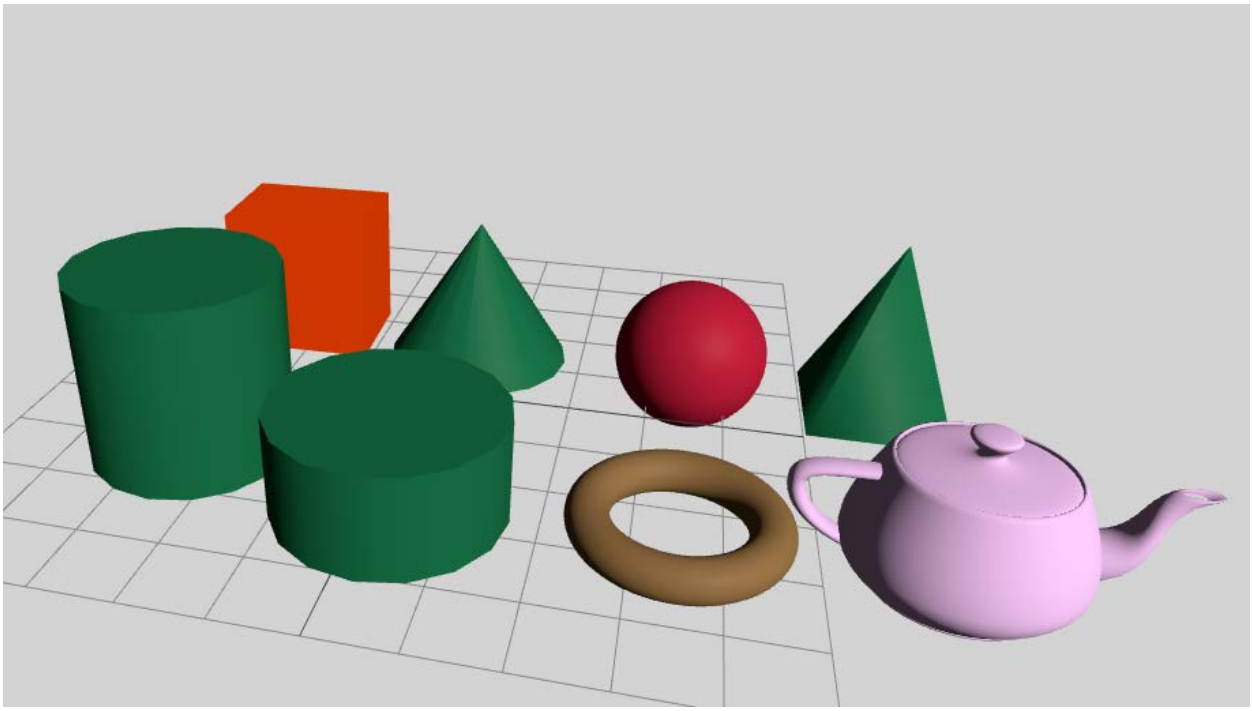

Рисунок 5 – Итоговый результат

Это лишь некоторые примеры того, что можно изобразить с помощью Three.js. Ознакомиться с результатом и кодом скрипта можно на сайте нашего проекта: https://almazismagilov.github.io/Figure/.

Вывод. В целом Three.js имеет обширные средства для процесса создания 3D-графики веб-браузером, делая этот процесс доступным для широкого круга разработчиков и открывая возможности для создания интерактивных и визуально привлекательных веб-приложений и игр.

Библиографический список:

1. Get started with WebGL [Электронный ресурс] / Microsoft. – Электрон. текстовые дан. – Режим доступа: http://msdn.microsoft.com/ru-ru/Library/dn385807(v=vs.85).aspx, свободный. – Загл. с экрана.
2. Three.js – JavaScript 3D library [Электронный ресурс] / Mr.doob. – Электрон.

текстовые дан. – Режим доступа: http://threejs.org, свободный. – Загл. с экрана.

3. Вильданов А.Н. 3D-моделирование на WebGL с помощью библиотеки Three.js: учебное пособие. - Уфа: РИЦ БашГУ, 2014. – 114 с. – ISBN: 987-5-7477-3560-6.

4. Касьяник В.В. Создание 3D-графики для веб-ресурсов средствами WEBGL и Three.js / В.В. Касьяник, И.Р. Лукьянович // Актуальные проблемы гуманитарного образования : Материалы IX Международной научно-практической конференции, Минск, 27–28 октября 2022 года / Редколлегия: О.А. Воробьева (гл. ред.) [и др.]. – Минск: Белорусский государственный университет, 2022. – С. 427-431.

Оригинальность 76%