**Лабораторная работа №2**

**Перемещение объектов в трёхмерном пространстве**

**Цель работы:**

- изучение способов перемещения объектов в трёхмерном пространстве с помощью Three.js;

- освоение основных методов расчёта координат объектов в трёхмерном пространстве.

**Задание:**

Необходимо разработать веб-приложение, представляющее собой упрощённую модель Солнечной системы. Модель должна включать следующие объекты:

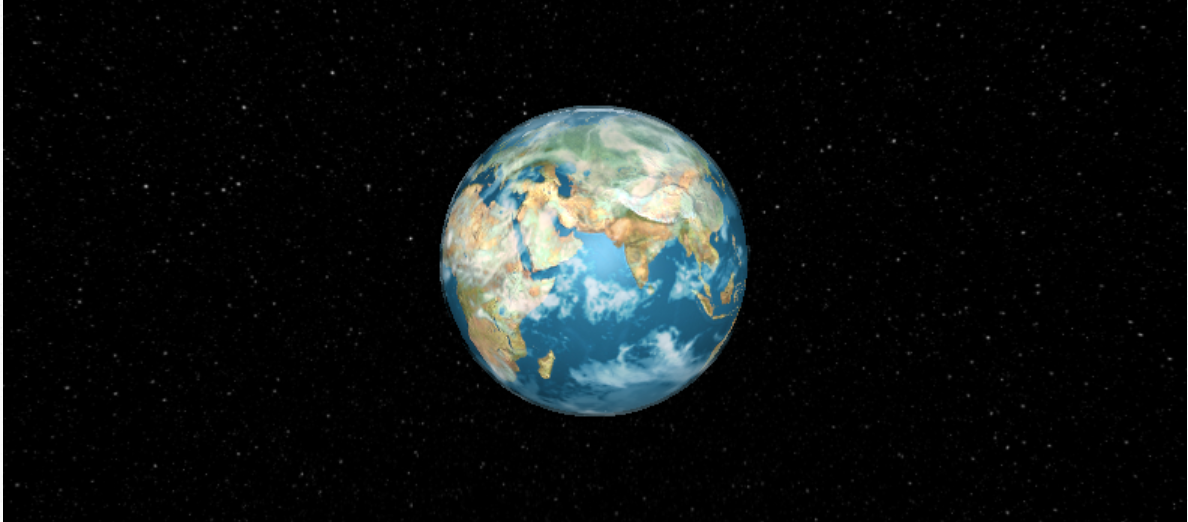
* карта звёздного неба;
* Солнце;
* Меркурий;
* Венера;
* Земля и Луна
* Марс

В качестве траекторий движения планет и Луны можно использовать окружность.

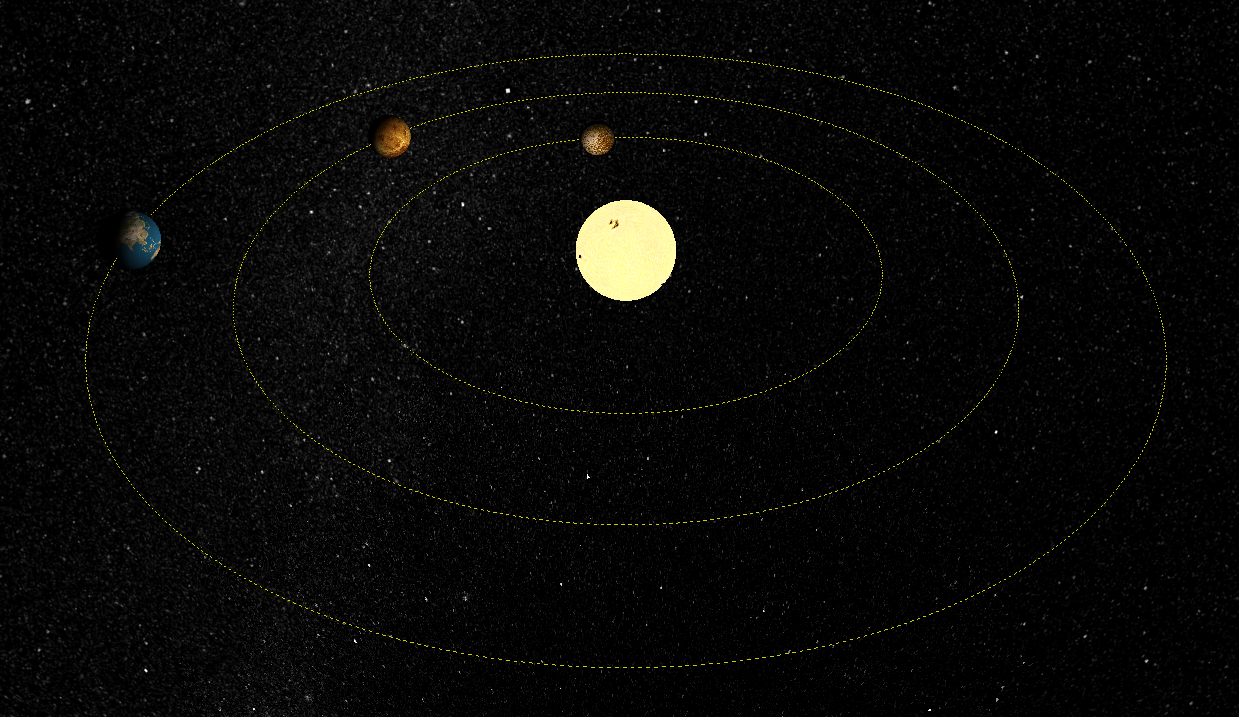
Расстояния между планетами, их скорость вращения вокруг собственной оси и солнца должны отражать реальные отношения размеров, расстояний и скоростей. (Марс меньше Земли, Меркурий меньше Марса и т.д.)

Также требуется реализовать режим слежения за планетами. По нажатию на клавиши 1 – 4 фокус камеры должен смещаться вслед за позицией планеты, соответствующей номеру нажатой клавиши. По нажатию кнопки 0 должен включаться общий вид на Солнечную систему.

Режим слежения:



Общий вид:



**Результаты выполнения задания представить в виде:**

* архив с проектом приложения. Если размер архива больше 2 Мбайт, то рекомендуется загрузить проект на <https://github.com/> (или на другое общедоступное хранилище) и предоставить ссылку;
* отчет по лабораторной работе в формате Microsoft Word, который содержит следующие разделы:

1. титульный лист;
2. задание на лабораторную работу;
3. описание разработанной программы и используемых алгоритмов со скриншотами выполнения;
4. вывод с результатами работы.

**Копирование исходного кода программ не допускается. Проекты с одинаковым исходным кодом зачитываться не будут.**

**Справочная информация**

К лабораторной работе приложены следующие дополнительные материалы:

* файлы библиотек **three.min.js** и **THREEx.KeyboardState.js**;
* необходимые графические файлы.

**Часть 1: Позиционирование объектов**

Библиотека **Three.js** предоставляет набор функций для создания простых моделей. Подобные модели удобно использовать для создания простых объектов. В частности, в данной лабораторной работе для создания планет понадобится сфера:

// создание геометрии для сферы

var geometry = new THREE.SphereGeometry(10, 32, 32);

// загрузка текстуры

var tex = loader.load("imgs/sunmap.jpg");

tex.minFilter = THREE.NearestFilter;

// создание материала

var material = new THREE.MeshBasicMaterial({

    map: tex,

    side: THREE.DoubleSide

});

// создание объекта

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

// размещение объекта в сцене

scene.add(sphere);

Параметры объекта THREE.SphereGeometry следующие:

* радиус сферы;
* число сегментов по горизонтали;
* число сегментов по вертикали.

Выражение tex.minFilter = THREE.NearestFilter означает, что используемая текстура будет автоматически отмасштабирована до ближайшей степени двойки.

Помимо текстуры, материал может содержать различные карты, отвечающие за визуальные характеристики объекта: рельеф, блики, отражения и др. Например, для использования карты рельефа, необходимо задать соответствующие параметры материала:

// загрузка карты рельефа

var bump = loader.load("imgs/earthbump1k.jpg");

// назначение карты рельефа и её масштабирования

var material = new THREE.MeshPhongMaterial({

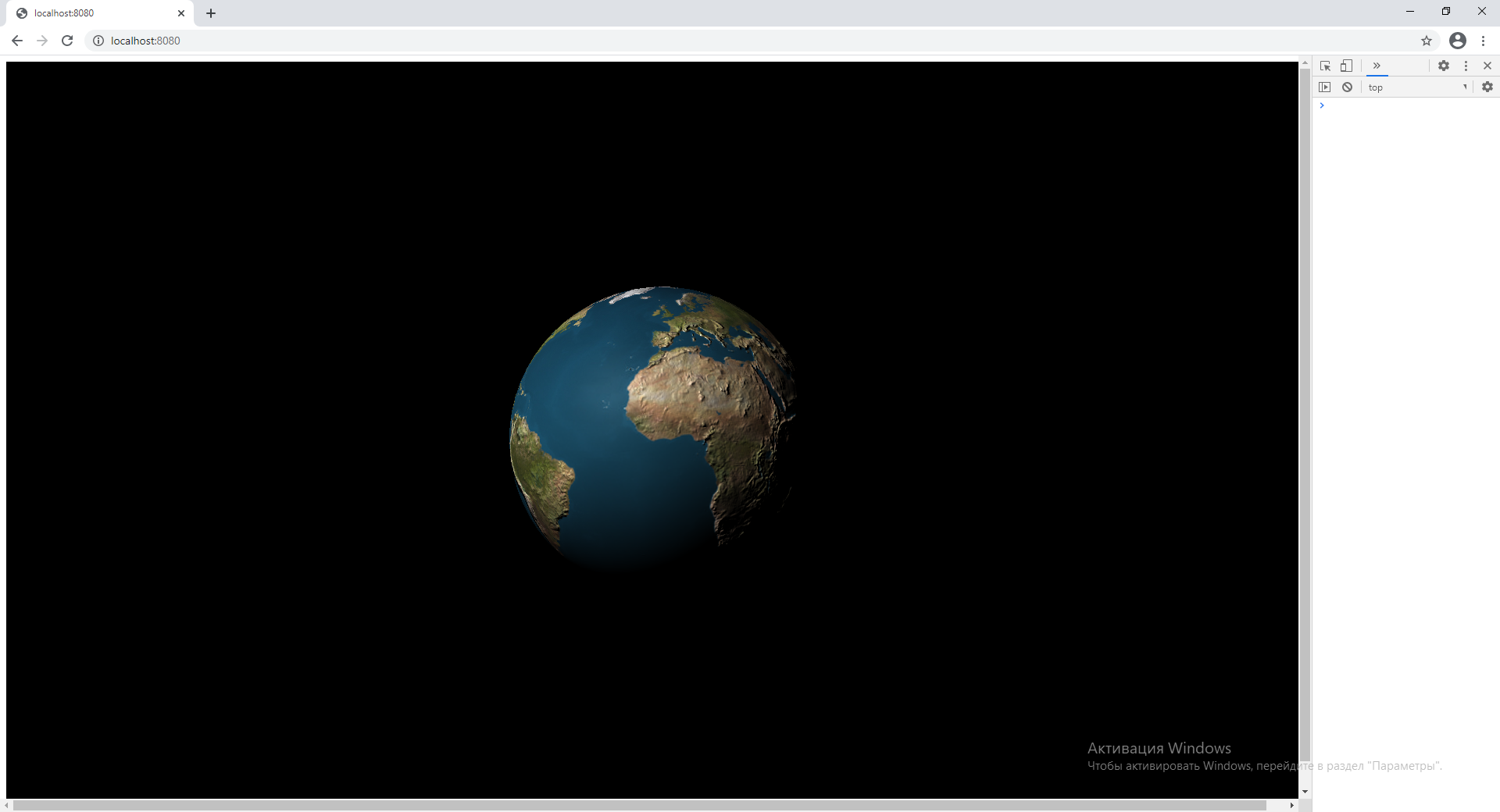
map: tex,

    bumpMap: bump,

    bumpScale: 0.5,

    side: THREE.DoubleSide

});



В остальном работа со сферой ни чем не отличается от работы с пользовательским объектом из предыдущей лабораторной.

Задать позицию для объекта можно при помощи свойства position одним из следующих способов:

// способ 1

sphere.position.x = x;

sphere.position.y = y;

sphere.position.z = z;

// способ 2

sphere.position.set(x, y, z);

// способ 3

var pos = new THREE.Vector3(x, y, z);

sphere.position.copy(pos);

Аналогичным образом при помощи свойства rotation для объекта может быть указан его поворот. Угол поворота указывается в радианах.

**Задание:** реализовать набор функций, позволяющих создавать объекты типа «Солнце», «планета», «звёздное небо». Все объекты должны содержать в себе информацию о текстуре, размере объекта и его начальной позиции. Помимо этого, объекты типа «планета» должны содержать информацию о собственном вращении и перемещении.

При помощи разработанных функций необходимо отобразить на экране неподвижную модель солнечной системы. Желательно соблюсти относительный масштаб планет и их удаление от Солнца.

Хранение планет необходимо реализовать в виде массива. Работа с массивом планет может быть организована следующим образом:

// создание массива планет

var planets = [];

// создание объекта планеты

var planet = {};

// добавление поля для хранения сферы

planet.sphere = sphere;

// добавление поля для хранения позиции

planet.pos = pos;

// добавление объекта планеты в массив

planets.push(planet);

// перебор планет

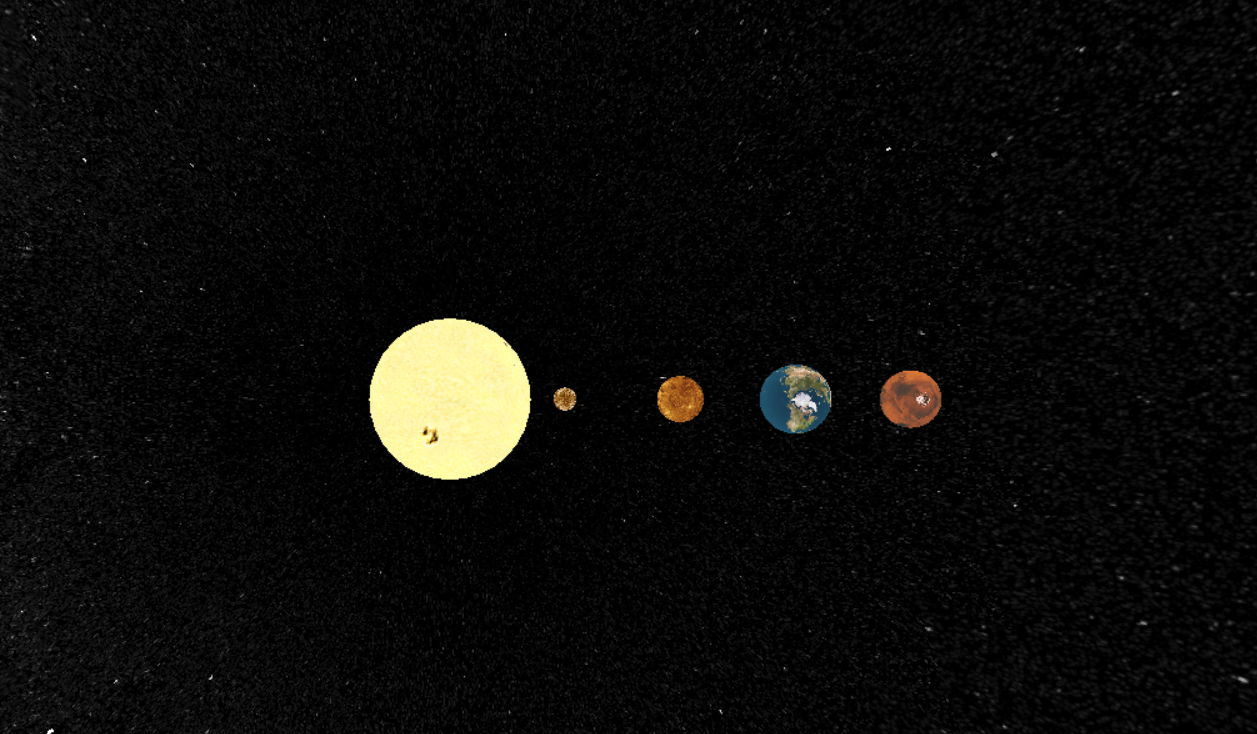
for (var i = 0; i < planets.length; i++)

{

    // действие

}

Пример (текстура звёздного неба наложена на большую сферу, внутри которой находится сцена):



**Часть 2: Перемещение объектов**

Одним из способов перемещения объектов в трёхмерном пространстве является аналитический расчёт текущей позиции объекта и размещении его при помощи свойства, рассмотренного в предыдущей части.

Например, движение по окружности с центром в произвольной точке можно записать через параметрическое уравнение окружности:

pos.x = center.x + r \* Math.cos(a);

pos.z = center.z + r \* Math.sin(a);

где pos – точка на окружности радиуса r с центром в точке center.

Вращение объекта можно осуществить, задав поворот вокруг осей XYZ при помощи углов Эйлера:

object.rotation.x = angleX;

object.rotation.y = angleY;

object.rotation.z = angleZ;

Другой вариант - использование преобразований поворота и переноса:

// перенос вдоль одной из локальных координатных осей

object.translateX(distance);

object.translateY(distance);

object.translateZ(distance);

// вектор axis задаёт ось Y

var axis = new THREE.Vector3(0, 1, 0);

// перенос вдоль оси, заданной вектором в локальных координатах

object.translateOnAxis(axis, distance);

// поворот вокруг оси, заданной в локальных координатах

object.rotateOnAxis(axis, angle);

Так же, получить текущую позицию объекта можно при помощи следующих методов:

// получение матрицы позиции из матрицы объекта

var m = new THREE.Matrix4();

m.copyPosition(object.matrix);

// получение позиции из матрицы позиции

var pos = new THREE.Vector3(0, 0, 0);

pos.setFromMatrixPosition(m);

Примечание: получить позицию в вектор можно напрямую из матрицы объекта.

Третий вариант - расчёт «матрицы мира» объекта при помощи комбинирования преобразований:

// создание набора матриц

var m = new THREE.Matrix4();

var m1 = new THREE.Matrix4();

var m2 = new THREE.Matrix4();

// создание матрицы поворота (вокруг оси Y) в m1 и матрицы перемещения в m2

m1.makeRotationY(angle);

m2.setPosition(new THREE.Vector3(x, y, z));

// запись результата перемножения m1 и m2 в m

m.multiplyMatrices(m1, m2);

// установка m в качестве матрицы преобразований объекта object

object.matrix = m;

object.matrixAutoUpdate = false;

В рамках выполнения данной лабораторной работы могут понадобиться следующие методы класса THREE.Matrix4:

// очистка данных

matrix.identity();

// умножение матрицы matrix на матрицу mat

matrix.multiply(mat);

// создание матрицы переноса в точку xyz

matrix.makeTranslation(x, y, z);

// создание матрицы поворота вокруг оси axis на угол angle (в радианах)

matrix.makeRotationAxis(axis, angle);

**Задание:** реализовать функцию,осуществляющую перемещение планет по круговым траекториям одним из выше указанных методов. Реализовать функцию отрисовки орбит планет пунктирной линией.

Чтобы скорость перемещения объектов не зависела от производительности компьютера, рекомендуется использовать функцию получения времени, прошедшего с предыдущего вызова этой функции:

var clock = new THREE.Clock();

var delta = clock.getDelta();//возвращает время, прошедшее с момента предыдущего вызова этой функции

Для отрисовки пунктирной линии рекомендуется использовать стандартный примитив:

var lineGeometry = new THREE.Geometry();

var vertices = lineGeometry.vertices;

// начало сегмента линии

vertices.push(new THREE.Vector3(x1, y1, z1));

// конец сегмента линии

vertices.push(new THREE.Vector3(x2, y2, z2));

// параметры: цвет, размер черты, размер промежутка

var lineMaterial = new THREE.LineDashedMaterial({

    color: 0xcccc00,

    dashSize: 1,

    gapSize: 1

});

var line = new THREE.Line(lineGeometry, lineMaterial);

line.computeLineDistances();

scene.add(line);

**Часть 3: Обработка нажатия клавиш и управление камерой**

Для обработки нажатия клавиш необходимо подключить библиотеку **THREEx.KeyboardState.js**:

<script src="js/libs/threex.keyboardstate.js"></script>

Например, можно проверить, нажата ли клавиша «0» в данный момент:

var keyboard = new THREEx.KeyboardState();

...

if (keyboard.pressed("0")) {

    // действие

}

Перемещение камеры осуществляется, так же, как и перемещение трёхмерных объектов.

**Список литературы:**

1. Эрик Фримен, Элизабет Робсон «Изучаем программирование на JavaScript»
2. Дэвид Флэнаган «JavaScript. Подробное руководство»
3. Сайт библиотеки Three.js <http://threejs.org/examples/>
4. Основы Three.js <https://threejsfundamentals.org/threejs/lessons/ru/>
5. Three.js Examples <http://stemkoski.github.io/Three.js/>