Липецкий государственный технический университет

Факультет автоматизации и информатики

Кафедра АСУ

ЛАБОРАТОРНАЯ РАБОТА №3

по организации графических систем и систем мультимедиа

Создание 3D-сцены

Студент Лапшова А.Г.

Группа М-АС-19

Руководитель Кургасов В.В.

Липецк 2020г.

Задание кафедры

Реализовать создание 3D-сцены (не менее 3-х объектов) на WebGL для отображения в браузере.

Цель работы

Освоить на практике создание 3D-сцены.

1 Теоретические сведения

1.1 Общие сведения

Для реализации программного продукта был выбран Babylon.js.

Babylon.js - фреймворк с открытым исходным кодом позволяющий создавать полноценные 3D-приложения и игры, работающие в браузере без каких-то дополнительных плагинов, что звучит очень здорово. Babylon.js включает в себя все необходимые инструменты для разработки 3D-приложений. Он позволяет загружать и использовать 3D-объекты, управлять ими, создавать и управлять специальными эффектами, воспроизводить и управлять звуками, создавать геймплей и многое другое. Babylon.js достаточно прост в применении [1].

Babylon.js создан с использованием языка TypeScript. TypeScript - компилируемый и мультиплатформенный язык генерирующий чистый JavaScript код.

Babylon.js достаточно большой проект, с момента его появления на GitHub он активно развивается по сей день. В нем создано большое количество функций с большим количеством параметров для большей гибкости. Язык TypeScript помогает разрабатывать более надежные и простые в сопровождении приложения, за счет типизации и ООП.

1.2 Возможности и недостатки Babylon.js

Основные возможности:

- Сцена: использование готовых мешей, туман, скайбоксы.
- Физический движок (модуль oimo.js).
- Сглаживание.
- Анимационный движок.
- Звуковой движок.
- Система частиц (партиклов).
- Аппаратное масштабирование.
- Поддержка LOD-ов.

- Пошаговая загрузка сцены.
- Автоматическая оптимизация сцены.
- Панель отладки.
- 4 источника освещения.
- Пользовательские материалы и шейдеры.
- Широкие возможности текстурирования.
- SSAO.
- Блики.
- 9 видов камеры, в том числе и для сенсорного управления.
- Экспортеры для 3ds Max, Blender, Unity3D, Cheetah 3d.
- Карта высот.

Основные типы камер, которые предоставляет рассматриваемый фреймворк:

- Свободная камера (free camera) – данную камеру можно перемещать по сцене, используя клавиши-стрелки на клавиатуре, а направление можно задавать с помощью мыши. Также можно включить необязательные гравитацию и обнаружение коллизий.
- Камера дугового поворота (arc rotate camera) – используется для поворота вокруг конкретной цели. С помощью событий от мыши, клавиатуры или сенсорных событий, пользователь может рассматривать объект со всех направлений.
- Сенсорная камера (touch camera) – свободная камера, использующая в качестве ввода сенсорные события. Подходит для любых мобильных платформ.
- Фиксированная камера (follow camera) – автоматически следует за конкретной целью (т. е. фиксируется на ней).

Как было сказано ранее, в Babylon.js имеется четыре типа освещения:

- Полусферическое – рассеянный свет (ambient light), в котором предопределены фоновый цвет (ground color) (пиксели внизу), цвет неба (пиксели вверху) и отражаемый цвет (specular color).
- Точечное (point) – свет, излучаемый из одной точки во всех направлениях подобно солнцу.
- Узконаправленное (spot) – как и предполагает название, это свет из одной точки с конкретным направлением и радиусом излучения. Этот свет может создавать тени.
- Направленное (directional) – свет, излучаемый в конкретном направлении откуда угодно. Солнце может быть точечным источником света, но направленное освещение лучше имитирует солнечный свет. Этот свет тоже может создавать тени, и именно его я использовал в своем примере.

Проблемы при использовании данного фреймворка: сцены, созданные с помощью Babylon.js очень требовательные к аппаратным ресурсам компьютера (хотя это утверждение применимо ко всем разработкам на базе WebGL). Также отсутствует подробная документация на некоторые «сложные» вещи [2].

2 Основная часть

2.1 Описание работы программы

Программа запускается по событию загрузки DOM-элементов на странице, обработчик данного события инициализирует 3D-сцену.

Первым этапом создания сцены является получение элемента canvas, для отображения созданных элементов. После создается движок BABYLON, а также происходит создание сцены и камеры, которая прикрепляется к элементу canvas. Также создается свет типа HemisphericLight, который имитирует свет окружающей среды.

Для импорта объекта из уже готовой модели, необходимо загрузить отдельную сцену, которая внутри (с использованием анонимной функции) управляет объектами. Импортируемому объекту «Космический корабль» устанавливаются такие параметры как позиция и орбита, а в методе registerBeforeRender происходит анимация передвижения по окружности.

Сцена «Космос» представляет собой набор нескольких объектов типа сферы, в центре которой находится объект «Солнце» с установленной текстурой, а также создан точечный объект света для реалистичности представления.

Создание трех планет происходит также с помощью объекта сферы фреймворка BABYLON. Устанавливается позиция, а также орбита по которой будет происходит движение созданной планеты.

«Звездное небо» создано с помощью объекта куб, внутри которого наложена текстура для реалистичности представления космоса.

Для понимания общей концепции передвижения планет на рисунке 1 отмечены radius, target(изначальное место создания объекта до перемещения позиции), а также alpha и beta окружности, по которым происходит передвижение. Каждый раз, при перерендеринге изображения пересчитываются параметры позиции x и z, что соответствует перемещению по alpha-окружности.

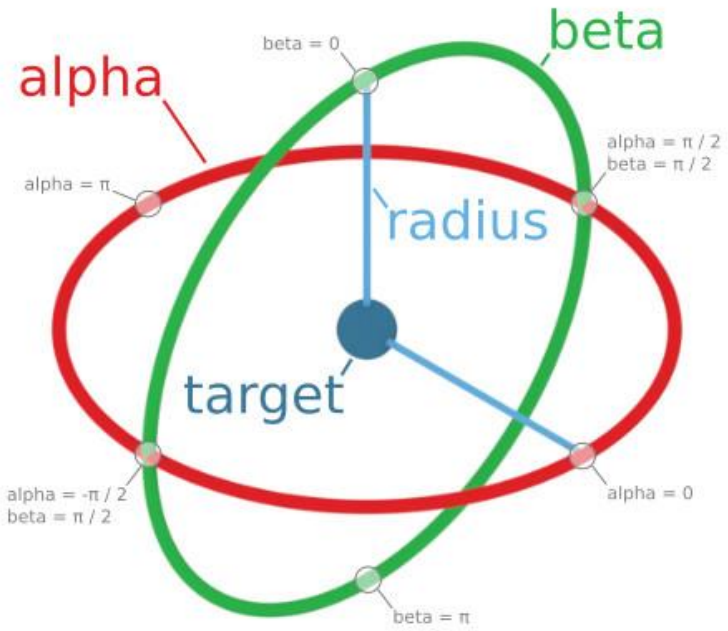

Рисунок 1 – Геометрия в BABYLON.js

2.2 Листинг кода

```
/// <reference path="lib/babylon.2.1.d.ts" />

var BjsApp = BjsApp || {};

BjsApp.init = () => {
    //Получаем элемент canvas
    const canvas = document.getElementById('renderCanvas');

    //Генерация движка BABYLON 3D
    const engine = new BABYLON.Engine(canvas, true);

    //Создание сцены
    const scene = new BABYLON.Scene(engine);

    //Добавление камеры и прикрекпление к эл-ту канвас
    const camera = new BABYLON.ArcRotateCamera(
        'camera',
        0,
        1.2,
        15,
        BABYLON.Vector3.Zero(),
        scene
    );
    camera.attachControl(canvas);

    camera.upperRadiusLimit = 50;

    //Добавление света
```

```
    const light = new BABYLON.HemisphericLight(
        'light1',
        new BABYLON.Vector3(0, 1, 0),
        scene
    );
    light.intensity = 0.5;
    light.groundColor = new BABYLON.Color3(0, 0, 1);

    scene.clearColor = new BABYLON.Color3(0, 0, 0);

    //Создание 3D объекта из импортируемой модели
    BABYLON.SceneLoader.ImportMesh("","./assets/models/","star-wars-vader-
tie-fighter.babylon", scene, function (meshes) {
        scene.createDefaultCameraOrLight(true, true, true);
        meshes.forEach(function(mesh) {
            mesh.position.x = 5;
            mesh.position.y = 1;
            //mesh.scaling = new BABYLON.Vector3(0.03,0.03,0.03);

            mesh.orbit = {
                radius: mesh.position.z,
                speed: -0.025,
                angle: 0
            };

        });

        //Добавление анимации
        scene.getMeshByName("group").rotationQuaternion = null;
        scene.registerBeforeRender(function () {
            const object = scene.getMeshByName("group");
            object.rotation.y += 0.005;

            object.position.x = object.orbit.radius *
Math.sin(object.orbit.angle);
            object.position.z = object.orbit.radius *
Math.cos(object.orbit.angle);
            object.orbit.angle += object.orbit.speed;
        });

    });

    //Создание объекта "солнце"
    const sun = BABYLON.Mesh.CreateSphere('sun', 16, 4, scene);
    //Добавление материала/текстуры к объекту
    const sunMaterial = new BABYLON.StandardMaterial('sunMaterial', scene);
    sunMaterial.emissiveTexture = new BABYLON.Texture(
        'assets/images/sun.jpg',
        scene
    );
    sunMaterial.diffuseColor = new BABYLON.Color3(0, 0, 0);
    sunMaterial.specularColor = new BABYLON.Color3(0, 0, 0);

    sun.material = sunMaterial;

    //Добавление света для реалистичности представления солнца
    const sunLight = new BABYLON.PointLight(
        'sunLight',
        BABYLON.Vector3.Zero(),
        scene
    );
    sunLight.intensity = 2;
```

```
    //Создание планет
    const planetMaterial = new BABYLON.StandardMaterial('planetMat',
scene);
    //Установка текстуры
    planetMaterial.diffuseTexture = new BABYLON.Texture(
        'assets/images/sand.jpg',
        scene
    );
    planetMaterial.specularColor = new BABYLON.Color3(0, 0, 0);

    //Создание планет с помощью объекта BABYLON сферы
    const planet1 = BABYLON.Mesh.CreateSphere('planet1', 16, 1, scene);
    //Изменение стартовой позиции и начальных параметров
    planet1.position.x = 4;
    planet1.material = planetMaterial;
    planet1.orbit = {
        radius: planet1.position.x,
        speed: 0.01,
        angle: 0
    };

    const planet2 = BABYLON.Mesh.CreateSphere('planet2', 16, 1, scene);
    planet2.position.x = 6;
    planet2.material = planetMaterial;
    planet2.orbit = {
        radius: planet2.position.x,
        speed: -0.01,
        angle: 0
    };

    const planet3 = BABYLON.Mesh.CreateSphere('planet3', 16, 1, scene);
    planet3.position.x = 8;
    planet3.material = planetMaterial;
    planet3.orbit = {
        radius: planet3.position.x,
        speed: 0.02,
        angle: 0.2
    };

    //Создание "коробки" космоса
    const skybox = BABYLON.Mesh.CreateBox('skybox', 1000, scene);
    const skyboxMaterial = new BABYLON.StandardMaterial('skyboxMat',
scene);

    //dont render what we cant see
    skyboxMaterial.backFaceCulling = false;

    //Движение "коробки" вместе с камерой
    skybox.infiniteDistance = true;

    skybox.material = skyboxMaterial;

    //Удаление рефлексии для того, чтобы свет не отражался на поверхности
"космоса"
    skyboxMaterial.diffuseColor = new BABYLON.Color3(0, 0, 0);
    skyboxMaterial.specularColor = new BABYLON.Color3(0, 0, 0);

    //Установка текстуры
    skyboxMaterial.reflectionTexture = new BABYLON.CubeTexture(
        'assets/images/space',
        scene
    );
    skyboxMaterial.reflectionTexture.coordinatesMode =
```

```
        BABYLON.Texture.SKYBOX_MODE;

    //Устанавливаем параметры анимации перед каждым рендерингом
    scene.beforeRender = () => {

        //Вращение вокруг своей оси
        sun.rotation.y += 0.005;

        //Передвижение по орбитам
        planet1.position.x = planet1.orbit.radius *
Math.sin(planet1.orbit.angle);
        planet1.position.z = planet1.orbit.radius *
Math.cos(planet1.orbit.angle);
        planet1.orbit.angle += planet1.orbit.speed;

        planet2.position.x = planet2.orbit.radius *
Math.sin(planet2.orbit.angle);
        planet2.position.z = planet2.orbit.radius *
Math.cos(planet2.orbit.angle);
        planet2.orbit.angle += planet2.orbit.speed;

        planet3.position.x = planet3.orbit.radius *
Math.sin(planet3.orbit.angle);
        planet3.position.z = planet3.orbit.radius *
Math.cos(planet3.orbit.angle);
        planet3.orbit.angle += planet3.orbit.speed;
    };

    //Зацкиливание рендеринга
    engine.runRenderLoop(() => {
        scene.render();
    });

    //При изменении размера в браузере происходит изменение размера внутри
канваса
    window.addEventListener('resize',  () => {
        engine.resize();
    });
};
```

3 Результат работы программы

Репозиторий проекта находится на GitHub [3].

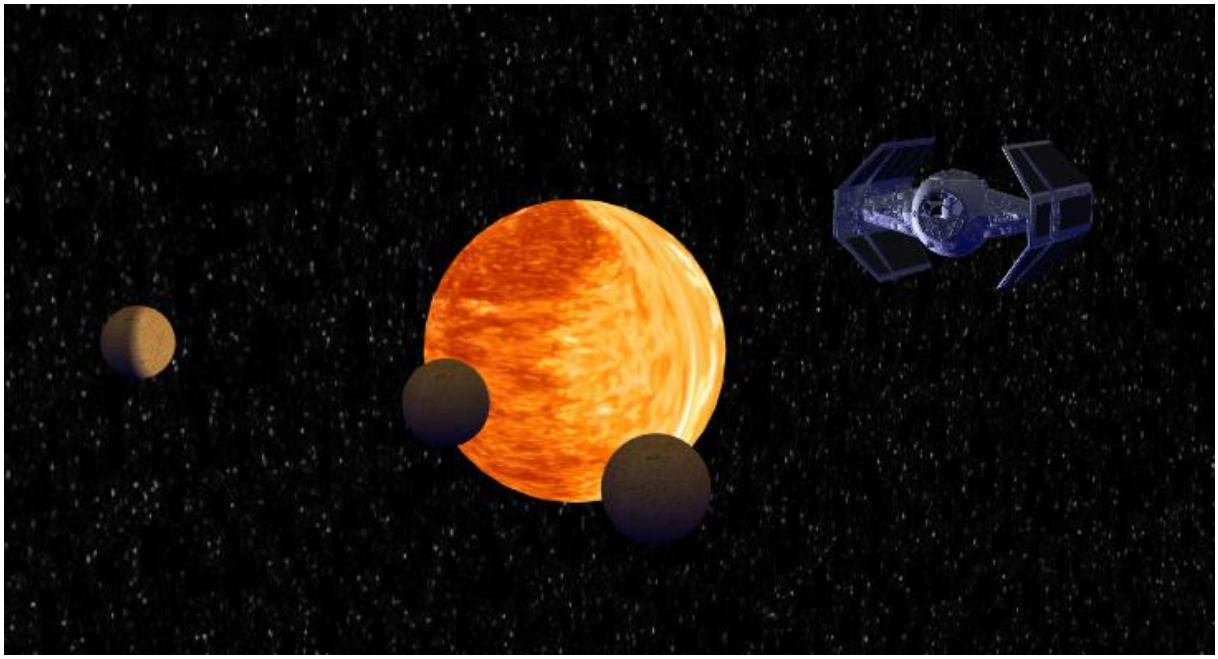

Рисунок 2 – 3D-сцена космоса

Вывод

В ходе выполнения данной лабораторной работы удалось реализовать создание 3D-сцены. Также были получены практические навыки обработки графики, а именно создание 3D-объектов с применением фреймворка Babylon.js. Изучены такие объекты, как камера, свет, текстуры, различные mesh-объекты, такие как шар и куб, а также получены навыки по импорту уже готовых объектов из 3D-редакторов.

Список источников

1. Babylon.js Documentation [Электронный ресурс]. – Электрон. текст. дан. – режим доступа: https://doc.babylonjs.com/, свободный.

2. Babylon.js [Электронный ресурс]. – Электрон. текст. дан. – режим доступа: https://ru.wikipedia.org/wiki/Babylon.js, свободный.

3. Репозиторий проекта [Электронный ресурс]. – Электрон. текст. дан. – режим доступа: https://github.com/nwdles/Graphics-and-Multimedia, свободный.