УДК 004.921 DOI: 10.17587/it.24.582-585

А. Д. Филинских, канд. техн. наук, доц., e-mail: alexfil@yandex.ru,
К. С. Корсаков, магистрант, e-mail: korsakov-kirill@mail.ru,
Нижегородский государственный технический университет им. Р. Е. Алексеева

Интерактивная трехмерная модель Нижегородского Кремля

Описана методика создания интерактивных трехмерных моделей на базе технологии WebGL, которая позволяет просматривать и управлять моделью в совместимых web-браузерах без сторонних плагинов. Создана сцена, описан способ добавления трехмерных моделей на сцену, добавления текстур моделям. Проведена работа с тенями для создания более реалистичной картины. Предлагается способ контроля камеры для оптимального просмотра модели. Предложены структура, процедура и функции для реализации проекта.

Ключевые слова: *трехмерная модель, WebGL, three.js, javascript, HTML, интерактивная трехмерная графика, Нижегородский Кремль*

Введение

Нижний Новгород — это город с богатой историей и большим культурным наследием. В городе имеется большое число памятников, исторических строений и красивых парков. На сегодняшний день демонстрация вышеперечисленных достопримечательностей реализуется за счет фото- и видеосъемки объектов и распространения информации посредством сети Интернет на различных ресурсах. Данный способ демонстрации не дает достаточной информации об объекте. Возникает задача повышения привлекательности города для туристов за счет разработки нового способа демонстрации достопримечательностей города с помощью технологии WebGL.

В данной работе описан процесс создания визуализации 3D-модели Нижегородского кремля с использованием технологии WebGL и библиотеки three.js.

WebGL [1] — это технология, базирующаяся на технологии OpenGL ES 2.0 и предназначенная для создания, а также отображения интерактивной двухмерной и трехмерной графики в веб-браузерах. Главной особенностью данной технологии является то, что для работы с ней не нужно устанавливать никаких сторонних плагинов или библиотек. Работа веб-приложений с использованием технологии WebGL основана на коде JavaScript, а некоторые элементы кода, так называемые шейдеры [2], могут выполняться непосредственно на графических процессорах [3], что значительно увеличивает производительность веб-приложений. Таким образом, для создания приложений разработчики могут использовать стандартные для веб-среды технологии HTML/CSS/JavaScript и при этом также применять аппаратное ускорение графики. Стоит отметить, что данная технология не привязана к какой-либо платформе и является кроссбраузерной. Также данная технология работает и на мобильных устройствах.

Перед написанием программного кода нужно подготовить трехмерную модель. В целом ее можно было бы выполнить и средствами WebGl, но данный способ нерационален, так как является достаточно трудоемким и требует большого количества времени. Наиболее просто и быстро это можно сделать, воспользовавшись трехмерным графическим редактором, например 3ds Max, Blender или ZBrush.

Основная часть

Интерактивная модель — это модель, с которой можно взаимодействовать, т. е. наблюдать ее реакцию и поведение при различных внешних воздействиях.

Интерактивная модель графического объекта должна обеспечивать:

- возможность изменения освещения, позволяющую рассмотреть модель со всех сторон;
- возможность управления положением модели в пространстве с помощью мыши для детального рассмотрения модели с разных сторон;
- возможность приближения и отдаления камеры;
- наличие текстур на моделях;
- размещение на хостинге для проверки на работоспособность и наглядной демонстрации.

Воспользуемся созданной на кафедре "Графические информационные системы" НГТУ им. Р. Е. Алексеева трехмерной моделью Нижегородского кремля и окружающего его рельефа, созданного на основе топографических карт в программе 3ds Max компании Autodesk.

Процесс построения интерактивной модели в технологии WebGl с помощью библиотеки three.js начинаем с создания основных элементов:

1) scene — место, где происходит визулизация [4];

2) camera — камера;

3) render — функция, отвечающая за визуализацию;

4) controls — применение контроля к камере;

5) light — источник света.

Создадим четыре направленных источника света типа **THREE.DirectionalLight** с разных сторон для того, чтобы модели были хорошо освещены со всех сторон.

После того, как основные элементы добавлены, начинаем работу непосредственно с моделями.

Добавление модели

В технологии WebGL двумя наиболее популярными загрузчиками являются **OBJLoader.js** и **OBJMTLLoader.js**. В ходе анализа загрузчика **OBJMTLLoader.js** было выявлено, что данный загрузчик добавляет модель целиком, а не отдельно каждую деталь, но возникли проблемы с отображением текстур, поэтому для работы был выбран загрузчик **OBJLoader.js,** так как он позволяет добавлять объекты на сцену поотдельности, а не целиком модель, тем самым увеличивая трудоемкость работы, но позволяя более тщательно настраивать их текстуры с помощи экземпляра **THREE.Texture**.

Так как модель может состоять и из одного, и из множества подобъектов, необходимо каждый подобъект модели, который состоит из геометрии и из первоначальной текстуры, сохранить в отдельную переменную (Mesh). Для этого создаем массив.

На примере модели ландшафта создаем загрузчик **objLoader**, в который подгружаем модель и прописываем функцию *console.log(object)*, позволяющую следить за процессом загрузки модели.

Далее прописываем функцию, которая пройдется по всем дочерним объектам и проверит типы данных. Если тип данных правильный, то он будет добавлен в созданный ранее массив:

```
object.traverse(function (child) {
    if (child instanceof THREE.Mesh)
          { meshesp.push(child); }});
```

Модель ландшафта состоит из одного дочернего элемента, поэтому создаем переменную и присваиваем ей единственный дочерний элемент, далее с помощью команды *scene. add(name)* добавляем переменную на сцену, заранее присвоив ей стандартный материал с помощью экземпляра стандартного материала **THREE.MeshNormalMaterial** (рис. 1, см. третью сторону обложки).

Добавление текстуры

Для того чтобы добавить текстуру на загруженную ранее модель, создаем менеджер загрузок (переменная — *manager*) с помощью экземпляра **THREE.LoadingManager**, после этого создаем загрузчик изображения (переменная — *loader*) с использованием экземпляра **THREE.ImageLoader** и передаем в него значение переменной *manager*. Создаем загрузчик текстур **THREE.Texture**, в который подгружаем сами текстуры и прописываем их параметры. Следует отметить, что для каждой текстуры необходимо создавать свой загрузчик текстур.

Присваиваем загруженную текстуру переменной дочернего элемента модели ландшафта с помощью экземпляра блестящего материала **THREE.MeshPhongMaterial** (рис. 2).

Блестящий материал был выбран, так как он является достаточно гибким для настройки и текстуры выглядят более реалистичными.

Процесс загрузки модели в технологии WebGL с помощью загрузчика моделей **OBJLoader** устроен так, что после загрузки объ-

екта нужно пройтись по каждому потомку, добавить его в массив, далее добавить его на сцену и после этого задать каждому потомку определенную текстуру. Так как дочерних элементов может быть несколько сотен, то добавления каждого из них — это достаточно трудоемкий и нерациональный процесс. Необходимо провести оптимизацию модели в программе 3ds Max, которая заключалась в объединении частей модели с одинаковыми текстурами в единое целое. Модель Архангельского собора до оптимизации состояла из 750 объектов, после оптимизации число дочерних элементов сократилось до 12 (рис. 3, *а*, *б*, см. третью сторону обложки).

Число строк кода неоптимизированной модели составляло 3800, после оптимизации число строчек кода сократилось до 54.

После оптимизации всех моделей переходим к загрузке и добавлению их на сцену описанным на примере ландшафта способом (рис. 4). Так как модели, сделанные в программе 3ds Max, при конвертации в формат.obj сохраняют координаты своего положения, то при добавлении их на сцену средствами WebGL они будут расположены на тех же местах, что и в трехмерной модели в 3ds Max.

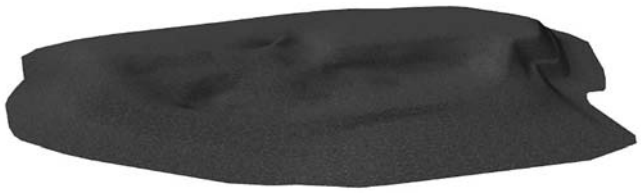

Рис. 2. Ландшафт с наложенной текстурой

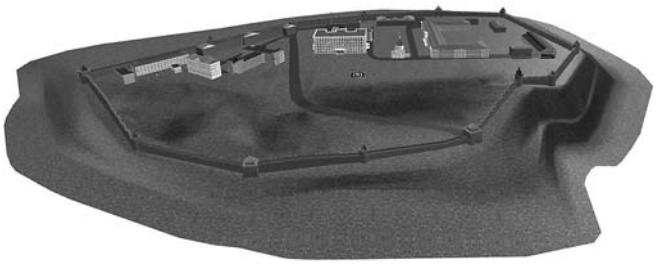

Рис. 4. Интерактивная модель Нижегородского Кремля

Рис. 6. Интерактивная модель Нижегородского Кремля на сайте

Добавление теней

Для придания реалистичности сцене библиотека three.js позволяет добавлять тень объектам.

Для того чтобы модели могли отбрасывать и принимать тень, необходимо в параметрах рендеринга параметру *shadowMap.enabled* и *shadowMap.soft* присвоить значение *true.*

С помощью параметров *castShadow* и *receiveShadow* каждому элементу на сцене добавляется возможность отбрасывать и принимать тень, для этого устанавливается значение *true* для этих параметров.

Следующим шагом является создание дополнительного источника света, благодаря которому объекты будут отбрасывать тень. Тень должна падать только в одну сторону, от одного источника света, поэтому добавляем пятый направленный источник света типа **THREE.DirectionalLight**, располагаемый дальше и выше остальных источников света. С помощью параметров *shadowDarkness*, *shadow.mapSize.width*, *shadow.mapSize.hight*, *shadow.camera.near*, *shadow.camera.far*, *shadow.camera.left*, *shadow.camera.bottom*, *shadow.camera.right*, *shadow.camera.top* настраиваем интенсивность теней, их качество и расстояние, на которое светит новый источник света.

После настройки теней сцена принимает вид, показанный на рис. 5 (см. третью сторону обложки).

Данная модель была выложена на сайт [5] (рис. 6).

Заключение

В результате работы рассмотрен процесс визуализации трехмерной модели Нижегородского Кремля с последующей публикацией в сети Интернет. Полученный результат был проанализирован в пяти самых популярных браузерах.

При открытии модели были выявлены следующие моменты:

1) геометрия моделей отображается без изменений;

2) имеется незначительное изменение в отображении текстур в двух браузерах;

3) основной функционал работает исправно (управление мышью, масштабирование).

Основываясь на полученном результате, можно сделать вывод о возможности использовать данную технологию для визуализации других трехмерных моделей различных достопримечательностей Нижнего Новгорода.

Список литературы

1. **WebGl**-технология. URL: https://metanit.com/web/webgl/1.1.php (Дата обращения 12.12.2016).
2. **Шейдеры**.URL:https://ru.wikipedia.org/wiki/%D0 %A8 %D0 %B5 %D0 %B9 %D0 %B4 %D0 %B5 %D1 %80 (дата обращения 9.10.2017).
3. **Графический** процессор. URL: http://we-it.net/index.php/zhelezo/videokarty/159-taktovaya-chastota-graficheskogo-protsessora-videokarty (дата обращения 6.10.2017).
4. **Визуализация**. URL: https://dic.academic.ru/dic.nsf/ruwiki/840376 (дата обращения 6.10.2017).
5. **Рабочая** модель. URL: http://v93080cg.beget.tech/

A. D. Filinskih, Ph. D., Associate Professor, e-mail: alexfil@yandex.ru,
K. S. Korsakov, Magistrant, e-mail: korsakov-kirill@mail.ru,
Nizhniy Novgorod State Technical University n. a. R. E. Alekseev, Nizhniy Novgorod, Russian Federation

Interactive Three-Dimensional Model of the Nizhny Novgorod Kremlin

The method is described for creating interactive three-dimensional models based on WebGL technology which allows to view and manipulate the model in compatible web browsers without other plug-ins. A scene is created, a method describes how to add three-dimensional models to the scene, add textures to models. Work has been done with shadows to create a more realistic picture. We propose a method of camera control, for optimal viewing of the model. The structure, procedure and functions for the project implementation are proposed.

Keywords: *three-dimensional model, WebGL, three.js, javascript, HTML, interactive three-dimensional graphics, Nizhny Novgorod Kremlin*

DOI: 10.17587/it.24.582-585

References

1. **WebGl**-tehnologija, available at: https://metanit.com/web/webgl/1.1.php (date of access 12.12.2016).
2. **Shejdery,** available at: https://ru.wikipedia.org/wiki/%D0 %A8 %D0 %B5 %D0 %B9 %D0 %B4 %D0 %B5 %D1 %80 (date of access 9.10.2017) (in Russian).
3. **Graficheskij** processor, available at: http://we-it.net/index.php/zhelezo/videokarty/159-taktovaya-chastota-graficheskogo-protsessora-videokarty (date of access 6.10.2017) (in Russian).
4. **Vizualizacija**, available at: https://dic.academic.ru/dic.nsf/ruwiki/840376 (date of access 6.10.2017) (in Russian).
5. **Rabochaja** model', available at: http://v93080cg.beget.tech/ (in Russian).