

Визуализация трёхмерной модели здания в веб-браузере на основе Three.js

Андриенко Иван Сергеевич

Приамурский государственный университет имени Шолом-Алейхема
Студент

Пащенко Екатерина Александровна

Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация

В работе рассматривается реализация одностраничного веб-приложения для визуализации трёхмерной модели здания в браузере. Приложение обеспечивает интерактивный просмотр 3D-модели с возможностью вращения, масштабирования, переключения этажей и отображения информационных подписей по помещениям. Вся трёхмерная часть реализована с использованием библиотеки Three.js и её дополнительных модулей для загрузки сжатых моделей формата GLB, управления камерой и отображения пользовательских аннотаций поверх сцены. Предложенное решение демонстрирует возможность использования браузерных технологий для наглядного представления архитектурных объектов без установки специализированного программного обеспечения.

Ключевые слова: трёхмерная визуализация; веб-приложение; Three.js; GLB; Draco; 3D-модель здания; интерактивная сцена; браузерный рендеринг.

Visualization of a three-dimensional model of a building in a web browser based on Three.js

Andrienko Ivan Sergeevich

Sholom-Aleichem Priamursky State University
Student

Paschenko Ekaterina Alexandrovna

Sholom-Aleichem Priamursky State University
Student

Abstract

The paper considers the implementation of a single-page web application for visualizing a three-dimensional model of a building in a browser. The application provides interactive viewing of a 3D model with the ability to rotate, scale, switch floors and display information signatures on the premises. The entire three-dimensional part is implemented using the Three library.js and its additional

modules for loading compressed GLB format models, controlling the camera, and displaying custom annotations on top of the scene. The proposed solution demonstrates the possibility of using browser technologies to visually represent architectural objects without installing specialized software.

Keywords: three-dimensional visualization; web application; Three.js; GLB; Draco; 3D model of the building; interactive scene; browser rendering.

1 Введение

1.1 Актуальность

Использование трёхмерных моделей зданий в информационных и презентационных системах позволяет повысить наглядность представления пространственных объектов и упростить восприятие архитектурной информации. Веб-реализации 3D-визуализации являются особенно востребованными, поскольку не требуют установки дополнительного программного обеспечения и доступны через стандартный браузер. Применение библиотек WebGL и Three.js обеспечивает возможность интерактивного взаимодействия с моделью и расширяет области использования подобных решений в архитектуре, градостроительстве и цифровых сервисах.

1.2 Обзор исследований

А.П. Митяева и Е.Н. Дроздова анализируют основные принципы создания трёхмерной графики в браузере на примере библиотеки Three.js, уделяя внимание структуре сцены и базовым средствам визуализации [1]. Д.И.Зуева рассматривает показатели производительности рендеринга трёхмерной графики с использованием WebGL и Three.js, выявляя ключевые факторы, влияющие на скорость отображения сцены [2]. А.Н. Вильданов и А.И. Исмагилов описывают возможности браузерного трёхмерного моделирования с применением Three.js, подчёркивая доступность и интерактивность веб-ориентированных 3D-решений [3]. В другой работе Д.И.Зуева исследует способы оптимизации производительности рендеринга 3D-графики в среде WebGL Three.js, акцентируя внимание на снижении вычислительной нагрузки и повышении стабильности отображения [4]. Mohan Balakrishnan Ja. и Manjunath Chavan S. рассматривают применение предварительной визуализации в процессе проектирования веб-анимаций на Three.js, отмечая её значение для планирования структуры и поведения трёхмерных сцен [5].

1.3 Цель исследования

Цель исследования – разработать и описать одностраничное веб-приложение для интерактивной визуализации трёхмерной модели здания в браузере с использованием библиотеки Three.js и вспомогательных модулей загрузки и управления сценой.

2 Материалы и методы

В качестве основы использовано одностраничное веб-приложение, реализованное с применением библиотеки Three.js для рендеринга 3D-сцены в браузере. Для загрузки модели здания в формате .glb применяются модули GLTFLoader и DRACOLoader, обеспечивающие поддержку сжатия Draco.

3 Результаты и обсуждения

Разработанное решение представляет собой одностраничное веб-приложение, предназначенное для интерактивного просмотра трёхмерной модели здания непосредственно в браузере. Пользователь получает возможность вращать модель, изменять ракурс обзора, переключать этажи и просматривать дополнительную информацию о помещениях. Вся логика работы с трёхмерной графикой сосредоточена в одном JavaScript-модуле, что упрощает сопровождение и развитие проекта, а интерфейсная часть вынесена в отдельные HTML- и CSS-файлы (рис. 1).

```
index.html > ...
1 |<!doctype html>
2 |<html lang="ru">
3 |
4 |<head>
5 |   <meta charset="utf-8" />
6 |   <meta name="viewport" content="width=device-width, initial-scale=1" />
7 |   <title>ЦОП – интерактивная модель</title>
8 |   <link rel="stylesheet" href="styles.css" />
9 |</head>
10|<body>
11|  <div class="backdrop"></div>
12|  <div id="app">
13|    <aside class="filters">
14|      <div class="brand">
15|        <span class="brand-mark">ЦОП</span>
16|        <span class="brand-sub">Интерактивная модель</span>
17|      </div>
18|      <div class="section">
19|        <h2>Режим просмотра</h2>
20|        <div class="filter-group">
21|          <button class="filter-btn active" data-filter="all">Все здание</button>
22|          <button class="filter-btn" data-filter="floor1">1 этаж</button>
23|          <button class="filter-btn" data-filter="floor2">2 этаж</button>
24|        </div>
25|      </div>
26|      <div class="section hint">
27|        <p>Выберите этаж, чтобы увидеть зоны кабинетов. Кликните на зону для деталей.</p>
28|      </div>
29|      <div class="section meta">
30|        <div class="meta-item">
31|          <span class="dot"></span>
32|          <span>Зоны подсвечиваются при наведении</span>
33|        </div>
34|        <div class="meta-item">
35|          <span class="dot alt"></span>
36|          <span>Модель можно вращать мышью</span>
37|        </div>
38|      </div>
39|    </aside>
40|    <main class="viewport">
41|      <div id="canvas-container"></div>
42|      <div class="hud">
43|        <div class="status" id="status">Загрузка модели...</div>
44|      </div>
45|    </main>
46|  </div>
47|</body>
48|</html>
```

Рис. 1. Структура проекта веб-приложения

На начальном этапе в приложении создаётся базовая 3D-сцена, включающая объект сцены, перспективную камеру и WebGL-рендерер. Камера настраивается таким образом, чтобы обеспечить удобный обзор модели здания, а рендерер инициализируется с включённым сглаживанием и прозрачным фоном. Такая конфигурация позволяет корректно отображать модель в различных браузерах и на устройствах с разным разрешением экрана (рис. 2).

```
50 const scene = new THREE.Scene();
51
52 const camera = new THREE.PerspectiveCamera(45, 1, 0.1, 500);
53 const renderer = new THREE.WebGLRenderer({ antialias: true, alpha: true });
54 renderer.setPixelRatio(Math.min(window.devicePixelRatio, 2));
55 renderer.outputColorSpace = THREE.SRGBColorSpace;
56 renderer.toneMapping = THREE.ACESFilmicToneMapping;
57 renderer.toneMappingExposure = 1.05;
58 renderer.setClearColor(0x000000, 0);
59 container.appendChild(renderer.domElement);
60
61 const labelRenderer = new CSS2DRenderer();
62 labelRenderer.domElement.style.position = 'absolute';
63 labelRenderer.domElement.style.top = '0';
64 labelRenderer.domElement.style.left = '0';
65 labelRenderer.domElement.style.pointerEvents = 'none';
66 container.appendChild(labelRenderer.domElement);
67
68 const controls = new OrbitControls(camera, renderer.domElement);
69 controls.enableDamping = true;
70 controls.enablePan = false;
71 controls.minDistance = 8;
72 controls.maxDistance = 180;
73 controls.autoRotate = true;
74 controls.autoRotateSpeed = 0.45;
75
76 const raycaster = new THREE.Raycaster();
77 const pointer = new THREE.Vector2();
78
79 const state = {
80   root: null,
81   floor1: null,
82   floor2: null,
83   roof: null,
84   zones: [],
85   activeFilter: 'all',
86   hovered: null,
87 };
```

Рис. 2. Инициализация сцены, камеры и рендерера

Для загрузки модели здания используется формат .glb, оптимизированный для передачи и отображения в веб-среде. Загрузка осуществляется с помощью GLTFLoader, а поддержка сжатия обеспечивается модулем DRACOLoader. Это позволяет существенно снизить размер передаваемых данных без потери визуального качества модели. После успешной загрузки объект добавляется в сцену и становится доступным для дальнейшего взаимодействия (рис. 3).

```
659 async function loadModel() {
660   setStatus('Загрузка модели...');
661
662   const loader = new GLTFLoader();
663   const draco = new DRACOLoader();
664   draco.setDecoderPath('https://www.gstatic.com/draco/v1/decoders/');
665   loader.setDRACOLoader(draco);
666
667   return new Promise((resolve, reject) => {
668     loader.load(
669       'COPP_zone.glb',
670       (gltf) => resolve(gltf),
671       (event) => {
672         if (event.total) {
673           const percent = Math.round((event.loaded / event.total) * 100);
674           setStatus('Загрузка модели... ${percent}%');
675         }
676       },
677       (error) => reject(error)
678     );
679   });
680 }
681
```

Рис. 3. Загрузка модели формата GLB с поддержкой Draco

Для обеспечения интерактивного взаимодействия с моделью применяется модуль OrbitControls. Он позволяет пользователю вращать сцену с помощью мыши, изменять масштаб и плавно перемещаться вокруг объекта. Дополнительно включён режим сглаживания движения камеры и автоматическое вращение модели, что повышает наглядность визуализации при пассивном просмотре (рис. 4).

```
68  const controls = new OrbitControls(camera, renderer.domElement);
69  controls.enableDamping = true;
70  controls.enablePan = false;
71  controls.minDistance = 8;
72  controls.maxDistance = 180;
73  controls.autoRotate = true;
74  controls.autoRotateSpeed = 0.45;
75
76  const raycaster = new THREE.Raycaster();
77  const pointer = new THREE.Vector2();
78
```

Рис. 4. Настройка управления камерой

Для вывода текстовой информации поверх трёхмерных объектов используется CSS2DRenderer. Данный механизм позволяет привязывать HTML-элементы к координатам 3D-сцены и отображать подписи к этажам или помещениям, не вмешиваясь в саму геометрию модели. Это делает интерфейс более информативным и удобным для пользователя.

После загрузки модели и инициализации всех компонентов запускается анимационный цикл, реализованный с помощью requestAnimationFrame. В рамках цикла обновляется состояние управления камерой и выполняется перерисовка сцены. Такой подход обеспечивает плавное взаимодействие с моделью и стабильное отображение результата при любых действиях пользователя (рис. 5).

```

693 function animate() {
694   requestAnimationFrame(animate);
695   controls.update();
696   renderer.render(scene, camera);
697   labelRenderer.render(scene, camera);
698 }
699
700 setupLights();
701 setupFilters();
702 resize();
703 window.addEventListener('resize', resize);
704 renderer.domElement.addEventListener('pointermove', handlePointerMove);
705 renderer.domElement.addEventListener('click', handleClick);
706 renderer.domElement.addEventListener('mouseleave', () => setHover(null));
707 closePanel.addEventListener('click', () => setPanelOpen(false));
708
709 loadModel()
710   .then((gltf) => {
711     state.root = gltf.scene;
712     scene.add(state.root);
713
714     const modelRoot = state.root.getObjectByName('ЦОПП') || state.root;
715     const findDirectChildByNames = (parent, names) => {
716       const list = Array.isArray(names) ? names : [names];
717       for (const name of list) {
718         const direct = parent.children.find((child) => child.name === name);
719         if (direct) return direct;
720       }
721       for (const name of list) {
722         const found = parent.getObjectByName(name);
723         if (found) return found;
724       }
725       return null;
726     };

```

Рис. 5. Анимационный цикл визуализации

В результате работы приложения пользователь получает интерактивную трёхмерную модель здания, доступную для просмотра в браузере без установки дополнительного программного обеспечения. Реализованный подход демонстрирует возможность использования Three.js для построения наглядных и функциональных систем визуализации архитектурных объектов, применимых в информационных, презентационных и образовательных веб-решениях (рис. 6).

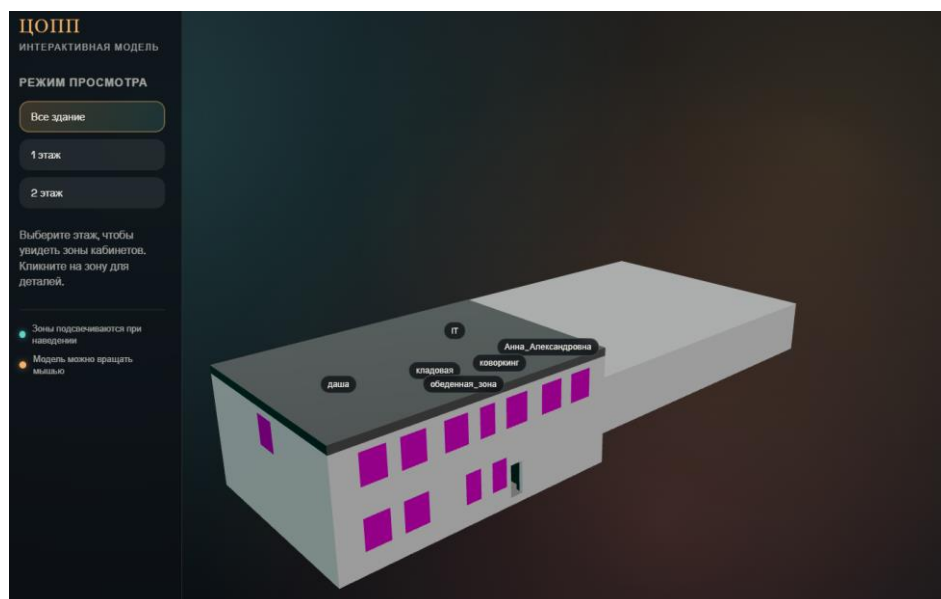


Рис. 6. Результат работы веб-приложения

4 Выводы

В ходе работы разработано одностраничное веб-приложение для интерактивной визуализации трёхмерной модели здания в браузере. Реализована загрузка и отображение модели в формате GLB с использованием библиотеки Three.js и поддержкой сжатия Draco, что обеспечивает оптимальный баланс между качеством визуализации и производительностью. Настроены средства управления камерой и навигации пользователя, позволяющие осуществлять просмотр модели с различных ракурсов, а также реализован механизм отображения информационных подписей поверх трёхмерных объектов. Применение анимационного цикла обеспечивает плавное взаимодействие со сценой и стабильный вывод результата. Разработанное решение подтверждает возможность эффективного использования веб-технологий для представления архитектурных объектов без применения специализированного программного обеспечения.

Библиографический список

1. Митяева А.П., Дроздова Е.Н. Анализ принципов создания 3D графики на примере библиотеки Three.js // Вестник молодых ученых Санкт-Петербургского государственного университета технологии и дизайна. 2021. № 3. С. 42–47.
2. Зуева Д.И. Исследование показателей производительности рендеринга трехмерной графики WebGL Three.js // Научно-технические инновации и веб-технологии. 2023. № 1. С. 64–69.
3. Вильданов А.Н., Исмагилов А.И. Трехмерное моделирование в браузере с помощью Three.js // Дневник науки. 2023. № 6 (78).
4. Зуева Д.И. Исследование способов оптимизации производительности рендеринга 3D-графики WebGL Three.js // Научно-технические инновации и веб-технологии. 2023. № 2. С. 27–32.
5. Mohan Balakrishnan Ja., Manjunath Chavan S. Programmers' perspectives on the use of previsualization in the design process of Three.js web animation // American Journal of Art and Design. 2022. Т. 7. № 1. С. 1.