Липецкий государственный технический университет

Факультет автоматизации и информатики

Кафедра Автоматизированных систем управления

Лабораторная работа №3

по курсу «Организация графических систем»

Система ознакомления с техникой Apple в браузере с использованием WebGL

Студенты Попов А.Д.

Группа М-АС-21

Руководитель Кургасов В.В.

доцент

Липецк 2022 г.

Задание

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

Теоретические сведения

WebGL (Web Graphics Library) - программная библиотека для языка JavaScript предназначенная для визуализации интерактивной трёхмерной графики и двухмерной графики в пределах совместимости веб-браузера без использования плагинов. WebGL приносит в веб трёхмерную графику, вводя API, который построен на основе OpenGL ES 2.0, что позволяет его использовать в элементах canvas HTML5.

WebGL исполняется как элемент HTML5 и поэтому является полноценной частью объектной модели документа (DOM API) браузера. Может использоваться с любыми языками программирования, которые умеют работать с DOM API, например, JavaScript, Rust, Java и другими. Все ведущие разработчики браузеров Google (Chrome), Mozilla (Firefox), и Apple (Safari), являются членами Khronos и реализуют WebGL в своих браузерах. За счёт использования низкоуровневых средств поддержки OpenGL часть кода на WebGL может выполняться непосредственно на видеокартах. WebGL — это контекст элемента canvas HTML, который обеспечивает API 3D графики без использования плагинов. Первая спецификация была выпущена 3 марта 2011 года. Современная версия 2.0 (несовместима с версией 1.0) доступна с 27 февраля 2017 года.

Поддержка WebGL присутствует в Firefox 4+, Google Chrome 9+, Opera 12+, Safari 5.1+ и Internet Explorer 11+. Однако помимо поддержки WebGL браузером, необходима также его поддержка графическим процессором клиента.

Three.js — легковесная кроссбраузерная библиотека JavaScript, используемая для создания и отображения анимированной компьютерной 3D графики при разработке веб-приложений. Three.js скрипты могут использоваться совместно с элементом HTML5 CANVAS, SVG или WebGL.

Three.js позволяет создавать ускоренную на GPU 3D графику, используя язык JavaScript как часть сайта без подключения проприетарных плагинов для браузера. Это возможно благодаря использованию технологии WebGL.

Реализация системы

Система реализована с использованием JavaScript фреймворка Vue.js 3. Для работы с WEB GL используется пакет three.js. В качестве хостинга используется GitHub Pages.

App.vue

```
<script setup>
  import IPhone13PM from './components/iPhone13PM.vue';
  import IPhone12P from './components/iPhone12P.vue';
  import IPhone5S from './components/iPhone5S.vue';
</script>

<template>
  <div class="app">
    <i-phone13-p-m class="object"/>
    <i-phone12-p class="object"/>
    <i-phone5-s class="object"/>
  </div>
</template>

<style>
  * {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: sans-serif;
  }

  .app {
    display: flex;
    flex-direction: row;
    align-items: center;
    justify-content: space-between;
    flex-wrap: wrap;
  }
</style>
```

MyThree.vue

```
<script setup>
  import * as THREE from 'three';
  import { GLTFLoader } from 'three/examples/jsm/loaders/GLTFLoader';
  // import { OrbitControls } from 'three/examples/jsm/controls/OrbitControls';
  import { onMounted, watch, defineProps, reactive } from 'vue';

  const props = defineProps({
    root: {
      type: String,
```

```
    required: true
  },
  cameraOptions: {
    type: Object,
    default: () => ({
      fov: 45,
      near: 1,
      far: 100
    })
  },
  lightOptions: {
    type: Object,
    default: () => ({
      color: 0xffffff,
      intensity: 1
    })
  },
  objectOptions: {
    type: Object,
    default: () => ({
      position: {
        x: 0,
        y: 0,
        z: -3
      },
      rotation: {
        x: 0,
        y: 0,
        z: 0
      }
    })
  },
  gltfUrl: {
    type: String,
    required: true
  }
});

let renderer = reactive(undefined);
let container = reactive(undefined);
let camera = reactive(undefined);
let scene = reactive(undefined);
let object = reactive(undefined);

watch(() => props.objectOptions, (currentValue) => {
  object.rotation.y = currentValue.rotation.y;
}, { deep: true });

const createRenderer = () => {
  renderer = new THREE.WebGLRenderer();
  container.append(renderer.domElement);
```

```
    renderer.setSize(container.offsetWidth, container.offsetHeight);
    renderer.setClearColor(0xffffff, 0)
  };

  const createCamera = () => {
    const { fov, near, far } = props.cameraOptions;
    camera = new THREE.PerspectiveCamera(fov, container.offsetWidth /
container.offsetHeight, near, far);
  };

  const createScene = () => {
    scene = new THREE.Scene();
  };

  const init = () => {
    createRenderer();
    createCamera();
    createScene();
    createObject();
    createLight();
    update();
  };

  const createObject = () => {
    const { position, rotation } = props.objectOptions;
    const loader = new GLTFLoader();
    loader.load(
      props.gltfUrl,
      gltf => {
        object = gltf.scene; scene.add(object);
        object.position.set(position.x, position.y, position.z);
        object.rotation.set(rotation.x, rotation.y, rotation.z);
        },
      () => {},
      error => {console.log(error)}
    )
  };

  const createLight = () => {
    const { color, intensity } = props.lightOptions;
    const light = new THREE.AmbientLight(color, intensity);
    scene.add(light);
  };

  const update = () => {
    requestAnimationFrame(update);
    renderer.render(scene, camera);
  };

  const onWindowResize = () => {
    camera.aspect = container.offsetWidth / container.offsetHeight;
```

```
    camera.updateProjectionMatrix();
    renderer.setSize(container.offsetWidth, container.offsetHeight);
  };

  onMounted(() => {
    container = document.querySelector(props.root);
    addEventListener('resize', onWindowResize);
    if (container) {
      init();
    }
  });
</script>
```

iPhone13PM.vue

```
<script setup>
  import MyThree from './MyThree.vue';
  import { reactive, ref } from 'vue';

  let rotationOptions = reactive({ x: 0, y: 0, z: 0 });
  let timeout = ref();

  const mouseDown = (add) => {
    timeout = setInterval(() => {
      if (add) {
        rotationOptions.y += 0.1;
      } else {
        rotationOptions.y -= 0.1;
      }
    }, 20);
  };

  const mouseClick = (add) => {
    if (add) {
      rotationOptions.y += 0.5;
    } else {
      rotationOptions.y -= 0.5;
    }
  };

  const mouseUp = () => {
    clearInterval(timeout);
  };
</script>

<template>
  <div class="iphone13pm">
    <a
      class="header"
      target="_blank"
```

```
      href="https://www.dns-shop.ru/product/8c6be93d15daed20/67-smartfon-apple-
iphone-13-pro-max-128-gb-goluboj/"
    >
      <h1>iPhone 13 Pro Max</h1>
    </a>

    <my-three
      root=".iphone13pm"
      gltfUrl="models/iPhone13PM/scene.gltf"
      :object-options="{ position: { x: 0, y: 0, z: - 3.5 }, rotation:
rotationOptions }"
      :light-options="{color: 0xffffff, intensity: 2 }"
    />

    <div class="buttons">
      <button
        class="button-left"
        @click="mouseClick(false)"
        @mousedown="mouseDown(false)"
        @mouseup="mouseUp"
      >
        <img src="@/assets/chevron-down.svg" style="width: 40px; height: 40px">
      </button>

      <button
        class="button-right"
        @click="mouseClick(true)"
        @mousedown="mouseDown(true)"
        @mouseup="mouseUp"
      >
        <img src="@/assets/chevron-down.svg" style="width: 40px; height: 40px">
      </button>
    </div>
  </div>
</template>

<style>
  .iphone13pm {
    width: 100%;
    height: 100vh;
    position: relative;
  }

  .buttons {
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    display: flex;
    flex-direction: row;
    align-items: center;
```

```
    justify-content: space-between;
    width: 70%;
  }

  .button-right {
    cursor: pointer;
    transform: rotate(-90deg);
  }

  .button-left {
    cursor: pointer;
    transform: rotate(90deg);
  }

  .header {
    position: absolute;
    top: 80%;
    left: 50%;
    transform: translate(-50%, -50%);
    text-decoration: none;
  }
</style>
```

Результат работы

Система позволяет просмотреть 3D модели iPhone 13 Pro Max, iPhone 12 Pro и iPhone 5s. Покрутить модели по оси Y с помощью стрелок и по нажатии на название перейти в магазин, где можно купить выбранную моделью

Рисунок 1 - Скриншот системы (iPhone 13 Pro Max)

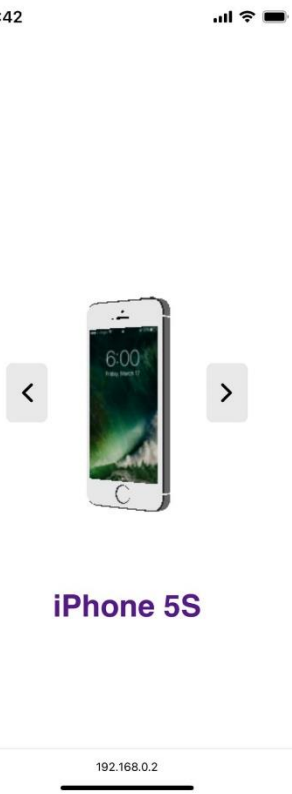

Рисунок 2 - Скриншот системы (iPhone 5s)

Вывод

В результате лабораторной работы была разработана система в виде веб-приложения которая позволяет просмотреть 3D модели айфонов и перейти в магазин для покупки выбранной модели.

Ссылка на приложение: https://drposeidon.github.io/lab3-threejs/

Ссылка на github: https://github.com/DrPoseidon/lab3-threejs