基于 WebGL 的 3D 可交互汽车展示系统设计与实现

作者：祝林海

【摘要】本文通过对基于互联网的交互式虚拟现实（Virtual Reality on Web,WebVR）和 Web3D 等相关方面的技术研究，针对现有汽车产品展示度不足的问题，提出一种基于 WebVR 的网页产品展示技术。该技术利用 Web 图形库 WebGL 技术、Three.js 框架和 Blender 建模技术架构建可交互汽车 3D 展示系统，实现了汽车产品的 3D 和 VR 展示。这种新的产品展示方式，在帮助汽车门户网站或车企进行汽车品牌宣传过程中，能够直观的展示汽车内饰和外饰，通过自定义汽车颜色和装配轮胎，自定义动画场景等可交互式操作，能更好地吸引客户，有利于商家进行品牌宣传和销售商品。

【关键词】虚拟现实、WebGL、Three.js、汽车展示系统

一、前言

虚拟现实（VirtualReality,VR），其实是介于虚拟与现实之间的一个载体，即将现实中的物体在计算机上进行虚拟化展示。利用其给人带来前所未有的新鲜感以及身临其境的代入感的特点，将其灵活运用到数字化展示平台上，就可以带来“沉浸式”的购物体验。　　Web3D 将虚拟现实技术应用到网页上，可以实现网页上的物品与用户之间的交互。以 HTML5 为代表的 Web 技术助推了 VR 在解决供给问题上的发展。Web3D 凭借 Web 开发者基数大、具有广大的受众以及可降低技术门槛的优点并利用 VR 的便捷性和轻巧性，运用于虚拟购物、数字化展示等创新领域。VR 内容展示以及交互都可以通过 Web 轻松实现，为 VR 带来了更加广阔的发展空间。

通过 Web3D 可以实现 3D 模型展示，这些 3D 模型应用在房产、家装行业、服装、工业建模等领域，能够带来更直观的视觉体验。随着技术的发展、基础网络的建设，Web3D 技术还能得到更广泛的应用。

二、关键技术研究

（一）WebGL 概述

WebGL（全称 Web Graphics Library）是一种 3D 绘图协议，这种绘图技术标准允许把 JavaScript 和 OpenGL ES 2.0（OpenGL 是最常用的跨平台图形库）结合在一起，通过增加 OpenGL ES 2.0 的一个 JavaScript 绑定，WebGL 可以为 HTML5 Canvas 提供硬件 3D 加速渲染，这样 Web 开发人员就可以借助系统显卡来在浏览器里更流畅地展示 3D 场景和模型了，还能创建复杂的导航和数据视觉化。显然，WebGL 技术标准免去了开发网页专用渲染插件的麻烦，可被用于创建具有复杂 3D 结构的网站页面，甚至可以用来设计 3D 网页游戏等等。

普通网页与使用 WebGL 的网页对比图，见图 1 所示：

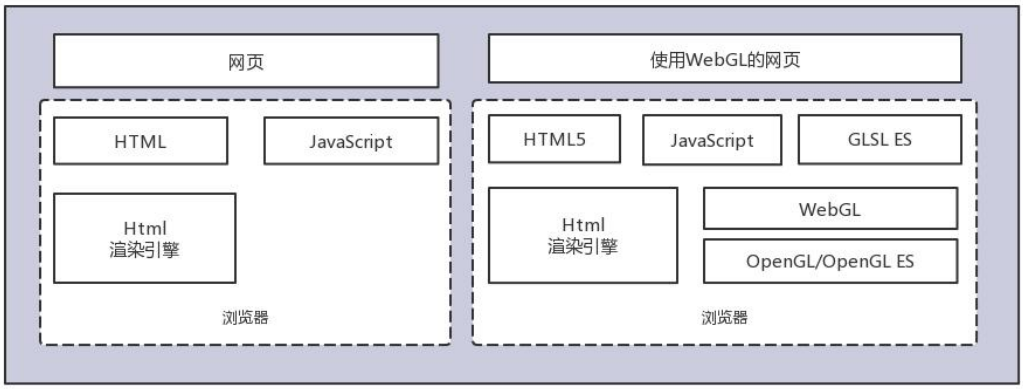

图 1

（二）WebGL 框架对比

基于 Web GL 的 3D 图形引擎有 Three.js、Babylon.js、Play Canvas、Cesium、Filament 等。本文重点对比 Three.js 和 Babylon.js。

1. Three.js

Three.js 是基于原生 WebGL 封装运行的三维引擎，在所有 WebGL 引擎中，Three.js 是国内文档资料最多、使用最广泛的三维引擎；Three.js 是纯渲染引擎，而且代码易读，适合作为学习 WebGL、3D 图形、3D 数学应用的平台，也可以做中小型的重表现的 Web 项目。除此之外，Three.JS 也有一定的缺点，比如部分重要功能不完善，与游戏相关的诸多功能需要二次开发等。

2. Babylon.js

Babylon.js 是一个简单但功能强大的 WebGL 驱动的 3D 图形引擎，它为 JavaScript 开发人员提供了简单易学的曲线，简单的 API 以及丰富的文档和教程列表。它可用于构建交互式 3D 展示/演示，3D Web 就绪的产品演示，游戏，VR（虚拟现实）应用程序以及复杂的体系结构仿真。Babylon.js 有很多开发人员社区，他们提供其代码片段，教程清单，当然还有可重用的扩展，以扩展其核心功能。

但是由于 Babylon.JS 封装程度高，使得加载模型所需的代码比 Three.JS 更少的同时也带来了拓展性下降，二次开发更加困难等缺点。

3. 总结

与 Babylon.js 相比，Three.js 在其库的扩展性，易用性以及功能方面有很好的优势。Three.js 国内文档资料最多、使用最广泛，进行 3D 开发不但门槛低，而且学习曲线不会太陡峭。同时，H5 和微信小程序中完全支持 Three.js 项目开发，并且已经有相关成功案例。所以，基于 Three.js 进行汽车展示系统开发是一个不错的选择。

（三）Three.js 框架分析

1. 基本组件

在 Three.js 中，有了场景（scene）、相机（camera）和渲染器（renderer） 这 3 个组件才能将物体渲染到网页中去。见图 2 所示：

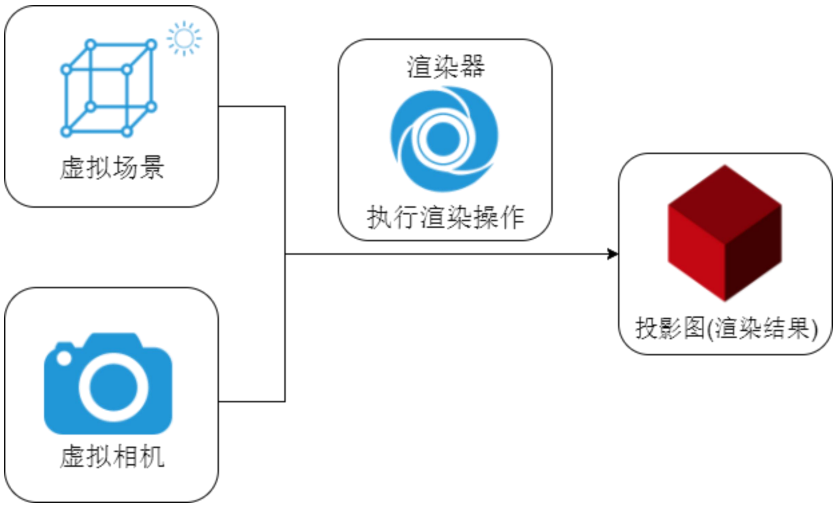

图 2

(1) 场景

场景是一个容器，可以看做摄影的房间，在房间中可以布置背景、摆放拍摄的物品、添加灯光设备等。

(2) 相机

相机 camera 的作用是定义可视域，相当于我们的双眼，生产一个个快照，最为常用的是 PerspectiveCamera 透视摄像机，其他还有 ArrayCamera 阵列摄像机（包含多个子摄像机，通过这一组子摄像机渲染出实际效果，适用于 VR 场景），CubeCamera 立方摄像机（创建六个 PerspectiveCamera（透视摄像机），适用于镜面场景），StereoCamera 立体相机（双透视摄像机适用于 3D 影片、视差效果）。

相机主要分为两类正投影相机和透视相机，正投影相机的话，所有方块渲染出来的尺寸都一样； 对象和相机之间的距离不会影响渲染结果，而透视相机接近真实世界，看物体会产生远近高低各不同。

(3) 渲染器

渲染器利用场景和相机进行渲染，渲染过程好比摄影师拍摄图像，如果只渲染一次就是静态的图像，如果连续渲染就能得到动态的画面。在 JS 中可以使用 requestAnimationFrame 实现高效的连续渲染。

3) 其他概念

Mesh 网格：有了场景和摄像头就可以看到 3D 场景中的物体，场景中的我们最为常用的物体称为网格。网格由两部分组成：几何体和材质。

材料（Materials），纹理（ Textures）：物体的表面属性可以是单纯的颜色，也可以是很复杂的情况，比如反射/透射/折射的情况，还可以有纹理图案。比如包装盒外面的贴图。

Geometry 几何形状：threejs 使用 Geometry 定义物体的几何形状，其实 Geometry 的核心就是点集，之所以有这么多的 Geometry，是为了更方便的创建各种形状的点集。光照（Lights）：组成部分。如果没有光源，我们就不可能看到任何渲染结果。一些常用的光源：

AmbientLight 环境光源，属于基础光源，为场景中的所有物体提供一个基础亮度。

DirectionalLight 平行光源：类似太阳光，发出的光源都是平行的。

HemisphereLight 半球光源：只有圆球的半边会发出光源。

PointLight 点光源：一个点向四周发出光源，一般用于灯泡。

SpotLight 聚光灯光源：一个圆锥体的灯光。

2. 模型格式

目前，3D 模型的格式有成千上万种可供选择，但每一种格式都具有不同的目的、用途以及复杂性。其中 WebGL 中常用的格式有 glTF 格式和 OBJ 格式。

glTF 格式

glTF（gl 传输格式）是一种开放格式的规范（open format specification），用于更高效地传输、加载 3D 内容。该类文件以 JSON（.glft）格式或二进制（.glb）格式提供，外部文件存储贴图（.jpg、.png）和额外的二进制数据（.bin）。一个 glTF 组件可传输一个或多个场景，包括网格、材质、贴图、蒙皮、骨架、变形目标、动画、灯光以及摄像机。glTF 是一种可以减少 3D 格式中与渲染无关的冗余数据并且在更加适合 OpenGL 簇加载的一种 3D 文件格式。

OBJ 格式

OBJ 文件是 AliaslWavefront 公司为它的一套基于工作站的 3D 建模和动画软件"Advanced Visualizer"开发的一种标准 3D 模型文件格式，很适合用于 3D 软件模型之间的互导。目前几乎所有知名的 3D 软件都支持 OBJ 文件的读写。OBJ 文件是一种文本文件，可以直接用写字板打开进行查看和编辑修改。OBJ 格式不包含动画、材质特性、贴图路径、动力学、粒子等信息。主要支持多边形(Polygons)模型。是最受欢迎的格式。

由于 glTF 这种格式是专注于在程序运行时呈现三维物体的，所以它的传输效率非常

高，且加载速度非常快。glTF 2.0 格式逐步的完成了 WebGL 的布局，也成为了这个领域的专用格式，随着发展游戏领域的应用也会越来越广泛。

3. 动画系统

创建动画的方式分为两种，一种是以补间动画的形式来指定实验对象的某个属性在规定时间内完成一定的变化，另一种是通过建模，设定模型对象通过骨骼动画的形式实现动态行为变化。

（1）补间动画

Tween.js 是一个补间动画引擎。即设定动画的起止状态，也就是关键帧，而关键帧之间的过渡状态由 Tween.js 计算。设定需要改变的元素的开始值和结束值，并设置好过渡时间，补间动画将会自动计算从开始到结束的状态，并产生平滑的动画变换效果。

（2）骨骼动画

在 Three.js 动画系统中，您可以为模型的各种属性设置动画：SkinnedMesh（蒙皮和装配模型）的骨骼，morph targets（变形目标），不同的材料属性（颜色，不透明度，布尔运算），可见性和变换。动画属性可以淡入、淡出、交叉淡化和扭曲。 在相同或不同物体上同时发生的动画的权重和时间比例的变化可以独立地进行。 相同或不同物体的动画也可以同步发生。

三、系统的设计与实现

（一）总体架构设计

根据汽车 3D 展示系统开发需求，本系统将视图模型层的业务逻辑功能划分为四个部分，具体结构见图 3 所示。

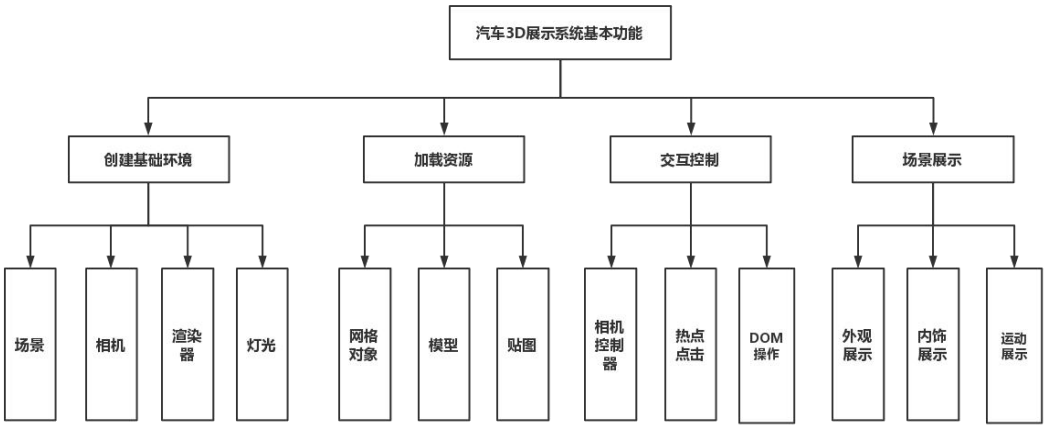

图 3

第一部分为创建基础环境，主要完成对展示系统的初始化。场景是整个展示系统的的载体，所有的物体都必须添加到场景中才能在相机的投影下渲染出来，相机决定了整个场景的视角，灯光负责照亮虚拟世界，而渲染器则将渲染结果输出到浏览器视显示。

第二部分为加载资源，在基础环境初始化后，还要根据系统展示功能加载相关资源。框架封装了常用的网格对象（Mesh），包括几何体（Geometry）和材质（Material）两部分，并提供了调用接口，开发者可以自由选择网格对象的几何形状以及材质。对于复杂的场景，框架还提供了加载器，支持从外部加载实验所需的模型（Model）和贴图（Texture），并完成自动装配。汽车模型是汽车 3D 展示系统最重要的组成部分。

第三部分是交互控制，汽车 3D 展示系统的可操作性直接决定用户的体验感。框架的交互分为三类，一是相机控制器，相机相当于人的眼睛，展示系统里的一切都是通过相机拍摄的，所以控制相机的视角就是改变场景，用户可以利用相机控制器设置场景的变换方式。二是模型选择，在汽车 3D 展示系统中可以添加各种自定义模型，通过光线投射 Raycaster（在三维空间中计算出鼠标移过了什么），触发模型的选择、高亮等交互行为。三是 DOM 事件，在 Web 系统中，可以通过响应鼠标或触摸事件，实现场景切换、模型动画等操作。

第四部分是场景展示，场景展示主要通过动画的形式展现。动画系统，在浏览器中一切的动态变化都是通过动画的形式实现的，从简单的关键帧到复杂的骨骼动画，框架提供了完整的动画系统，开发者将对象添加到动画系统，然后设置相应的属性值即可完

成动画的控制。

（二）创建基础环境

本系统通过 PerspectiveCamera 创建透视相机，通过场景中 fog 属性添加迷雾效果，控制整体场景的可视范围。添加的灯光包括环境光、点光源、顶部光源、尾部光源、以及模拟太阳炫光。其中通过点光源、顶部光源和尾部光源，三点布光用于降低场景模型的阴影效果。

（三）加载资源

在系统启动过程中，涉及的耗时资源包括模型、纹理图、库文件、音效文件等。为避免用户网络因素导致的白屏时间较长，会增加一个带加载进度的开屏页面。开屏页面的资源进度百分比通过 LoadingManager 来管理。

首先，在 Blender 或者 3Ds Max 等三维软件中，进行 3D 建模、贴图制作、动画制作等操作后，输出 glTF 格式模型文件，或者 OBJ 模型和相关的纹理贴图、法线贴图等。使用 Blender 进行汽车模型创建，并选定格式输出。见图 4 所示。

图 4

其次，在获得 glTF 模型文件后，采用 DRACOLoader 进行模型的压缩，调用了 GLTFLoader 加载器来解析模型，并将它们加载到场景中。根据模型信息，自动将贴图赋值到相应的材质上，最后创建网格对象将模型和材质组合输出，呈现完整的模型效果。

整体模型加载流程，见图 5 所示：

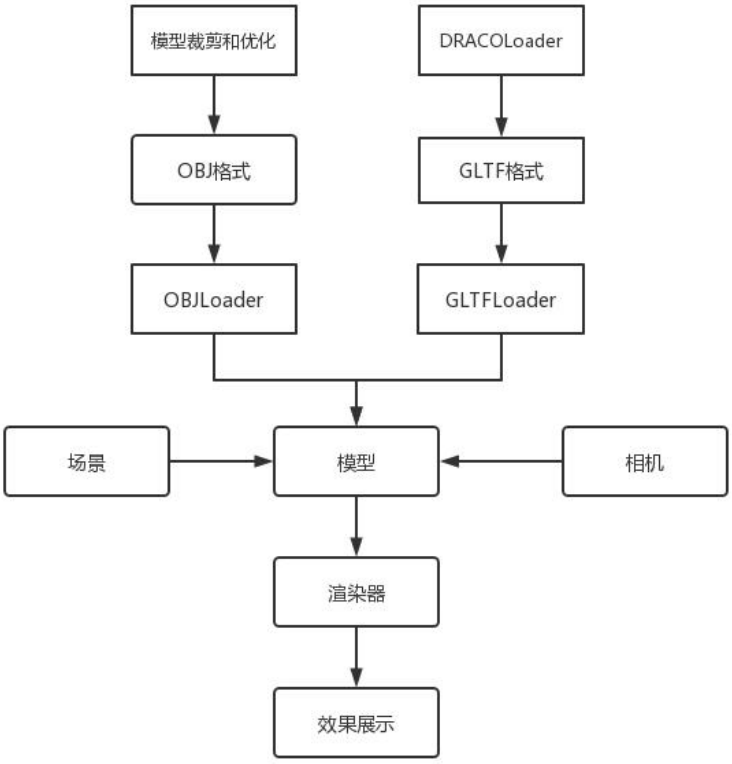

图 5

最后，使用 PlaneGeometry 创建平台的地面骨架。加载地面纹理图，基于 50x50 的方格，并循环展开，实现地面的搭建。其他资源包括天空场景的搭建，开、关车门和引擎音效播放，以及系统相关的纹理图。

（四）交互控制

本系统涉及的控制包括相机控制，模型点击事件，DOM 点击事件。

相机控制：通过添加 OrbitControls 控制器，可以实现场景的 720 度控制。实际项目中，为了避免穿透地平面，会设置 maxPolarAngle 属性来限制控制器的拖动范围，比如

限制为 Math.PI /2。

模型事件：通过 Raycaster 光线投射，可以确认鼠标点击或触摸的模型。通过点击的模型，可以响应自定义展示内容。比如门把手处添加热点，实现开关车门；在多媒体屏幕处添加热点，实现音视频播放等。

DOM 点击事件：通过 DOM 事件，可以在框架外部控制模型内容的变化。比如控制车轮动画启动，开关车门，改变汽车模型的颜色等。

（五）场景展示

系统场景场景主要分为外观展示，内饰展示，以及车身动态展示。

外观展示：主要是通过相机控制，进行汽车的前、后、左、右、顶部 5 个方向展示。通过自定义颜色，实现汽车的车身、车轮毂和玻璃颜色的变换。

内饰展示：主要通过设置当前相机中心为前驾驶坐或者后排，可以进行环绕 720 度展示，以及放大进行细节展示。

车身动态展示：基于模型动画和补间动画，可以实现开关车门，后背箱开关，车轮滚动，以及多场景动态展示。

搭建的系统界面，见图 6 所示：

图 6

四、总结与展望

本文通过对互联网的交互式虚拟现实和 Web3D 等相关方面的技术研究，实现了基于 WebGL 的 3D 可交互汽车展示系统设计与实现。3D 展示系统通过直观的展示汽车内饰和外饰，自定义汽车颜色和装配轮胎，自定义动画场景等可交互式操作，能更好地吸引客户，有利于商家进行品牌宣传和销售商品。

本文仅针对汽车 3D 汽车场景模型进行研究和实现，并未涉及更高精度模型和复杂场景。复杂场景和高精度模型，意味着更高的计算量，对用户的硬件计算能力和网络提出新的要求。

随着下一代 Web 的 3D 图形 API 标准“WebGPU”的提出，WebGPU 能够更好的提供现代 3D 图形和计算能力。相信基于 WebGPU 能够解决大量复杂场景和高精度模型的系统问题，进一步促进 Web3D 的开发和应用。

参考文献

[1]尹千慧,贺鹏飞,王玺联,孟令增,王海洋.基于 WebGL 的 3D 立库可视化系统设计与实现[J].信息技术,2021,(03):84-88.DOI:10.13274/j.cnki.hdzj.2021.03.016

[2]JosDirksen. Three.js 开发指南:the JavaScript 3D library for WebGL[M].北京: 机械工业出版社, 2015.

[3]李福送,王文军,林伟健,豆璇凯,王雪峰,苏社鑫.基于 WebGL 技术的机电产品 3D 在线交互展示实现[J].装备制造技术,2020(09):191-193+197.

[4]王孟博,董泽,石轲,苏子凡.WebGL 技术探索及几种基于 WebGL 的引擎比较[J].中国科技信息,2021(05):89-90.

[5]张晓琳.基于 B/S 模式的 3D 服装定制系统设计与实现[J].软件导刊,2018,17(02):96-98+101.