基于 WebGL 的交互平台设计与实现*

汪 浩 田 丰 张文俊
(上海大学影视艺术技术学院 上海 200072)

摘 要：介绍了 WebGL 技术及其应用框架 Three.js，描述了交互平台基本架构搭建和各个功能模块的设计与实现过程，并针对 OBJ 格式的 3 维模型提出一个模型加载算法解析模型的顶点、面和法向量等数据，然后把三维模型加载到网页平台中来，最后本文对平台进行了跨浏览器测试和模型载入速度测试，测试结果表明系统不需修改任何代码就能直接运行在 Firefox、Chrome 和 Opera 这 3 大主流浏览器上，并且模型载入速度较快，显示效果流畅。

关键词：WebGL；Three.js；模型加载；交互设计
中图分类号：TP368.2 **文献标识码**：A **国家标准学科分类代码**：510.50

Design and implementation of interactive platform based on webGL

Wang Hao Tian Feng Zhang Wenjun
(Institute of Film & TV Arts and Technology, Shanghai University, Shanghai 200072, China)

Abstract: We introduce the WebGL technology and Three.js which is one of WebGL application framework, we also describe the design and implementation of basic structure and each module of interactive platform, and propose an algorithm of loading 3D model to load the OBJ model by parsing data of model's vertex, face, normal vector and so on. At last we do the mult-browser testing and loading speed testing, the result shows that this platform can run correctly on Firefo, Chrome and Opera without modifying any code, its loading speed is fast and the platform can display the model smoothly.

Keywords: WebGL; Three.js; loading model; interactive design

1 引 言

在过去的十年里，互联网技术有了突飞猛进的发展，逐渐参透到人们生活的各个角落，使人们的生活方式产生了巨大的改变[1]。在互联网的各个领域中，发展和变化最快的就是 Web 应用的发展，已经成为当今网络技术的研究重点[2]。随着人们对网页体验的要求不断提高，网页正在逐渐地从传统的二维平面网页向交互式三维网页发展。

但是，早期的技术并不成熟，比如 Java Applet 所实现的非常简单的 Web 交互三维图形程序，不仅需要下载一个巨大的支持环境，而且画面非常粗糙，性能也很差，其主要原因就在于 Java Applet 在进行图形渲染时，并没有直接利用到图形硬件本身的加速功能[3]。

后来，Adobe 的 Flash Player 浏览器插件和微软 Silverlight 技术相继出现，逐渐成为了 Web 交互式三维图形的主流技术。与 Java Applet 技术不同，这两种技术直接利用操作系统提供的图形程序接口，来调用图形硬件的加速功能，从而实现了高性能的图形渲染。但是，这两种解决方案都存在一些问题。首先，它们是都是通过浏览器插件形式实现的，这就意味着对于不同的操作系统和浏览器需要下载不同版本的插件；其次，对于不同的操作系统，这两项技术需要调用不同图形程序接口。这两点不足在很大程度上限制了 Web 交互式三维图形程序的使用范围。

普通的 Web 浏览器正日益成为能应用丰富 3D 交互式程序的一个平台[4]。2014 年 10 月，万维网联盟完成了 HTML5 的标准制定，而作为 HTML5 标准之一的 WebGL 很好地解决了上述两个问题：首先，它通过 JavaScript 脚本实现 Web 交互式三维图形制作程序的设计与实现，无需任何浏览器插件支持；其次，它利用底层图形硬件的加速功能进行的图形渲染，是通过统一的、标准的、跨平台的 OpenGL ES2.0 实现的。

利用 WebGL 技术构建三维交互平台并加载三维模型，实现鼠标与键盘和网页三维场景中的模型进行互动。目前，国内关于 WebGL 和 HTML5 的研究还是比较少的，本文可以为后续研究者带来一定借鉴意义。

2 WebGL 技术及 Three.js 介绍

WebGL 是基于 OpenGL ES 2.0 的一种新的 API，在

收稿日期：2015-01
* 基金项目：上海市科委“科技创新行动计划”高新技术领域(14511108202)项目

浏览器中与 Web 页面的其他元素可以无缝连接；WebGL 具有跨平台的特性，可以运行在从手机、平板到家用电脑的任何主流操作系统当中[5]。WebGL 运行时其内部主要元素之间的关系如图 1 所示。

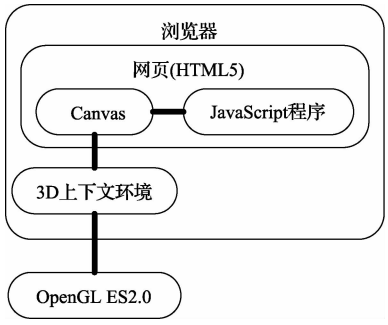

图 1　WebGL 中关键元素的连接关系

如图 1 所示，浏览器打开包含 Canvas 元素和 JavaScript 程序的网页，然后获取 3D 上下文环境（3D context）对象，该对象可以作为在 Canvas 元素上绘制 3D 场景的接口，并且还可以把 OpenGL ES2.0 和浏览器连接在一起。其中，Canvas 是 HTML5 引入的一个新元素，该元素可以在页面上指定位置实时渲染 2D 图像和 3D 图像，开发者不仅可以通过 HTML5 CanvasAPI 在 Canvas 画布上显示各种形状、渲染文字或者直接显示一幅图片，还可以自行操作画布颜色，并对其进行旋转，调整色彩透明度和填充色等，这些操作都是通过 JavaScript 程序实现的。

但是，直接基于 WebGL 开发交互式三维网页应用是非常困难的，需要对 OpenGL、三维设计以及 WebGL 的原生 API 非常了解，而且在实现过程中也极易出错。为了解决这些问题，一些开发人员开发出了许多基于 WebGL 的开源框架，从而为其他开发人员开发类似的程序提供了便利。目前，比较流行的 WebGL 框架有 Three.js、PhiloGL、Babylon.js、SceneJS 等。每个框架都有自己的特点。其中，Three.js 是一个轻量级的用于在浏览器中创建 3D 计算机图形图像应用程序的 JavaScript 库[6]。本文选用 Three.js 来开发系统，主要是因为 Three.js 是目前应用最广泛的 WebGL 框架，其文档资料也是最丰富的，而且完全采用 JavaScript 编写而成，非常适用于三维网页的开发[7]。

3　交互系统构建

3.1　基本环境搭建

Three.js 必须依托网页才能发挥作用，所以系统建立在基本 HTML 结构之上，HTML 基本结构如图 2 所示。主程序在 HTML 的 body 里面的 script 标签里面进行编写。

在把 Three.js 包含进行基本结构中之后，需要对三维环境进行设置。主要设置内容如下：

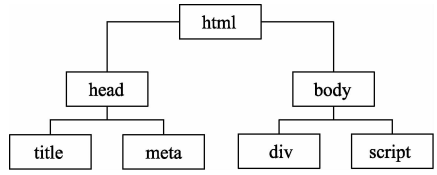

图 2　HTML 基本结构

1）渲染器设置

为了达到更好的展示效果，本系统选用 Three.js 中渲染效果较好的 WebGLRenderer，其兼容性不是最优，但是也可以满足要求。

2）相机设置

Three.js 提供了两种相机类型，即透视相机和正交相机[8]，由于本系统目的之一是提供真实感，故采用透视相机。

3）设置场景和光源

三维场景使用 THREE.Scene() 来设置；Three.js 中提供了 4 种基本光源，即环境光、点光源、聚光灯和平行光，系统中主要用到环境光、点光源和聚光灯，通过使用 THREE.AmbientLight()、THREE.SpotLight() 和 THREE.DirectionalLight() 来设置。

3.2　加载外部模型

三维模型是由三角形面或者四边形面组成的网格模型。在 Three.js 中用 THREE.Mesh 来表示网格模型。THREE.Mesh 的构造函数是：THREE.Mesh = function（geometry，material）。其中第 1 个参数 geometry 是一个 THREE.Geometry 类型的对象，包含了模型顶点之间的连接关系；第 2 个参数 Material 定义了模型的材质，材质会影响光照、纹理对 Mesh 的作用效果。本系统显示的 3D 模型事先用 Blender 做好，然后导出为 OBJ 格式，导出时还会附带导出 MTL 格式的材质

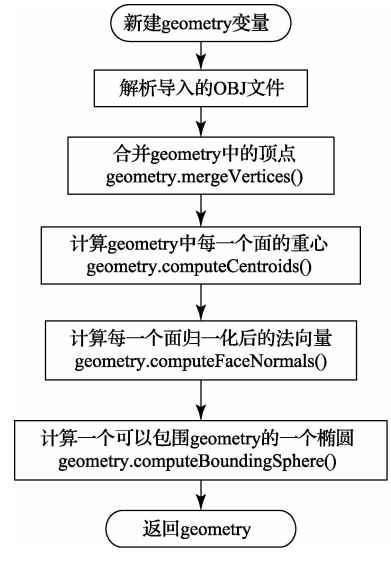

图 3　加载 OBJ 文件的程序流程

文件。本文提出一个模型加载算法,对三维模型文件进行解析,最后导入到网页中来。

OBJ 文件的加载算法流程图如图 3 所示。

其中最主要的是第 2 步解析导入的 OBJ 文件,算法中通过使用正则表达式对 OBJ 文件里面的文本信息进行解析和处理,然后把顶点、法向量、UV 坐标和面等索引信息转变成 geometry 里的顶点索引、法向量索引、UV 坐标索引以及面索引。

图 4 为 OBJ 模型用文本方式打开后的部分内容展示,图片中每一行表示的是三个顶点的坐标信息,程序读取该部分信息时所使用的正则表达式为:

/v(+[\d|\.|\+|\-|e|E]+)(+[\d|\.|\+|\-|e|E]+)(+[\d|\.|\+|\-|e|E]+)/;

该正则表达式可以读取“字母 v+空格+浮点型数据+空格+浮点型数据+空格+浮点型数据”这一类的文本信息。

```
2292  v -1.867902 -1.709669 -7.092330
2293  v -1.866299 -1.697875 -7.103549
2294  v -1.861550 -1.686535 -7.114338
2295  v -1.853840 -1.676083 -7.124281
2296  v -1.843463 -1.666922 -7.132996
2297  v -1.830819 -1.659404 -7.140148
2298  v -1.816393 -1.653818 -7.145463
2299  v -1.800741 -1.650378 -7.148735
2300  v -1.768185 -1.650378 -7.044102
2301  v -1.784463 -1.649216 -7.045207
```

图 4　OBJ 文件中的顶点信息

图 5 显示的为 OBJ 文件中的面信息,每一行显示的是顶点索引和对应的纹理坐标索引,例如“f 1304/11 1305/11 1337/11 1336/11”表示这个面是四边形面,其顶点索引为 1304、1305、1337 和 1336,对应纹理坐标索引为 11。对于面,算法中采用的正则表达式为:

/f(+(-? \d+)\/(-? \d+))(+(-? \d+)\/(-? \d+))(+(-? \d+)\/(-? \d+))(+(-? \d+)\/(-? \d+))? /

```
3011  f 1304/11 1305/11 1337/11 1336/11
3012  f 1283/11 1285/11 1323/11 1322/11
3013  f 1275/11 1276/11 1278/11 1277/11
3014  f 1319/11 1320/11 1352/11 1351/11
3015  f 1253/11 1255/11 1308/11 1307/11
3016  f 1277/11 1278/11 1280/11 1279/11
3017  f 1308/11 1309/11 1341/11 1340/11
3018  f 1275/11 1277/11 1319/11 1318/11
3019  f 1279/11 1280/11 1282/11 1281/11
```

图 5　OBJ 文件中的面信息

解析模型后,再按顺序调用 geometry 的 4 个方法,最终得到完整的 geometry 对象。把 MTL 文件转成 Three.js 里面的材质(material)文件,实现过程与加载 OBJ 文件类似,使用正则表达式对 MTL 文件里面的文本信息进行解析和处理,最终得到完整的 material 对象。得到 geometry 和 material 之后,再新建一个 mesh 变量并引入 geometry 和 material,如下面代码所示。

```
var mesh = new THREE.Mesh( geometry, material );
```

这样得到的 mesh 就可以渲染在网页中了。系统加载一辆汽车模型到网页中的效果图如图 6 所示。

图 6　模型加载完成后的显示效果

3.3　交互设计

三维空间中的交互主要使用鼠标和键盘,鼠标和键盘的触发事件通过添加事件侦听来实现,其核心代码如下:

```
document.addEventListener(“事件名称”, 函数, false);
function 函数(event) {
//函数体
}
```

本系统中鼠标实现的功能为:鼠标滚轮实现镜头的缩放(Zoom),按住鼠标左键并移动鼠标可以实现镜头摇摄(Pan),按住鼠标右键不放实现镜头的移动(Move)。

网页中鼠标的事件名称有 mousemove、mousedown、mouseup 和 mousewheel。分别为这 4 个事件编写对应触发函数,其中 mousemove 事件要和 mousedown 与 mouseup 相配合,例如,当产生 mousedown 事件和 mousemove 事件时,首先判断是鼠标左键还是右键,如果是左键,则函数功能实现镜头的摇摄,如果是右键,则实现镜头的移动。滚轮事件对应的触发函数则实现镜头缩放。总体示意图如图 7 所示。

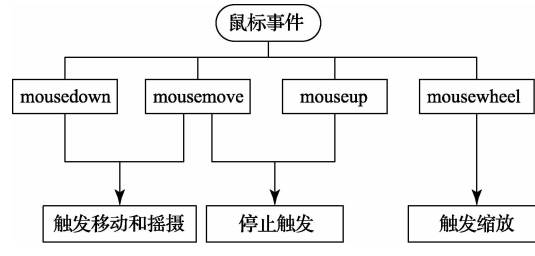

图 7　鼠标交互设计

网页中键盘的事件名称为 keydown 和 keyup，本平台中键盘事件主要用来控制模型的移动，用到 W 键和 S 键，触发函数的参数是 HTML5 的全局变量 event，这个 event 有一个属性 keyCode，通过判断 keyCode 与对应按键的 ASCII 码值是否相等来执行相应的指令，即 W 键前进，S 键后退。其中 W 按键模型运动算法流程图如图 8 所示，S 键触发的模型运动过程与 W 键一样，只是方向相反。该算法模拟真实世界中物体从静止到加速启动，然后保持匀速，最后减速停止这一过程。

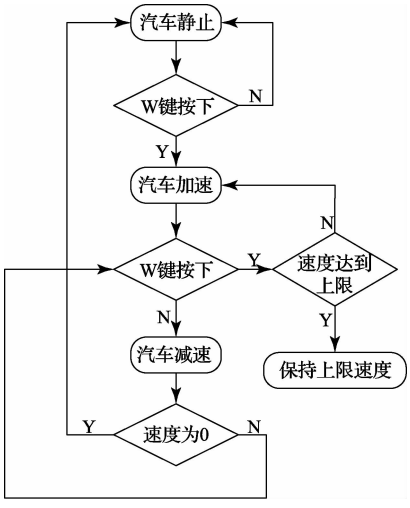

图 8　模型运动算法流程

由于系统基于浏览器发送请求，服务器接受请求并给与响应的机制，针对交互式系统的特点，开发模式采用 B/S 结构的开发模式[9]。

4　性能分析

本系统运行的 PC 环境：

1）CPU 为 Intel（R）Core（TM）2 DUO E8400 @ 3.0GHz；

2）内存为 2.00G；

3）显卡：英伟达 Geforce8499GS 显卡；

4）操作系统：Windows7 专业版 32 位操作系统。在此环境里对系统进行跨浏览器测试和模型载入时间测试。

4.1　跨浏览器测试

本次测试过程中不修改任何代码，直接运行在 3 大主流浏览器上，结果表明都可以正常运行；模型载入后，对模型的绘制与渲染是基于本地显卡，用户对模型进行旋转、缩放等操作均无任何卡顿、卡现象，十分流畅[10]。效果如图 9 所示。

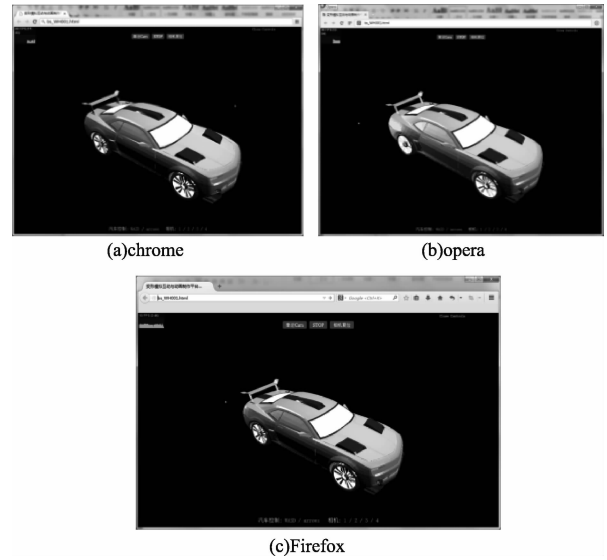

图 9　跨浏览器测试

4.2　模型载入时间测试

本文首先以汽车模型为例，对不同浏览器载入相同模型的响应时间进行了实验。汽车模型共由 11 253 个三角面单元构成，顶点数为 5 623 个。由于响应时间并不能保证精确一致，本文在采取 3 次载入相同模型后取平均值的方式进行测试，实验结果如表 1 所示。

表 1　不同浏览器载入响应时间实验

运行平台	浏览器	版本号	响应时间
PC Windows7	Chrome	3.0.2171.95	1.20
	Opera	26.0.1656.60	1.78
	Firefox	34.0.5	1.04

从表 1 可以看出，平台载入模型的时间较短，而且还能看出目前 Firefox 浏览器的表现最好。

然后以 Firefox 浏览器为运行环境，对不同模型大小下平台的响应时间进行测试，结果如表 2 所示，可以看出随着渲染面数增加，载入时间会增大，但是对模型的旋转、缩放等操作一直很流畅，用户体验性未受到影响。

表 2　不同模型大小的载入响应时间测试

渲染面数	响应时间/ms
313	223
1 056	467
5 487	605
11 253	1 040
50 745	1 806

（下转第 128 页）