文章编号:1671-4598(2020)02-0216-04 DOI:10.16526/j.cnki.11—4762/tp.2020.02.045 中图分类号:TP391.9 文献标识码:A

# 基于 Three.js 的飞行仿真系统设计

冯 姣,刘志勤,黄 俊,黎茂锋,杨 茂
(西南科技大学 计算机科学与技术学院,四川 绵阳 621010)

**摘要**:传统的富客户端飞行仿真系统在使用上受到应用程序和硬件设备的限制,导致其开放性、易用性、跨平台性存在一定缺陷,难以满足设计人员在离开实验室后的使用需求;针对上述问题,对 Web 三维可视化进行深入研究,利用 WebGL 在不安装任何渲染插件的情况下,支持浏览器端进行 2D/3D 硬件加速渲染的优势,以 WebGL 第三方图形库 Three.js 为实现基础,构建数据驱动的三维飞行可视化仿真系统,使用户可以在 Web 端无缝访问系统的服务资源;该系统由气动数据驱动,实时模拟了飞行器运动轨迹,同时兼顾天空效果、尾焰特效以及摄像机漫游,增加仿真效果真实性,使用户可以通过浏览器端得到直观丰富的三维仿真效果,具有良好的应用价值。

**关键词**:Web 三维可视化;WebGL;Three. js;数据驱动;三维仿真

# Design of Flight Simulation System Based on Three.js

Feng Jiao, Liu Zhiqin, Huang Jun, Li Maofeng, Yang Mao
(Southwest University of Science and Technology, Mianyang 621010, China)

**Abstract**: The traditional convenience rich client flight simulation system is limited by the application program and hardware equipment, which leads to Limitation of its openness, and cross-platform application, and it is difficult to meet the needs of designers after leaving the laboratory. In view of the above problems, in-depth study of Web 3D visualization, using WebGL to support the advantages of 2D/3D hardware accelerated rendering in the browser without installing any rendering plug-ins, based on WebGL third-party graphics library Three.js, build a data-driven 3D flight visualization simulation system, so that users can seamlessly access to the system's service resources on the Web. Driven by aerodynamic data, the system simulates the trajectory of the aircraft in real time, taking into account the sky effect, tail flame effect and camera roaming, which increases the authenticity of the simulation effect, and enables users to get intuitive and rich three-dimensional simulation results through the browser, which has good application value.

**Keywords**: Web 3D visualization; WebGL; Three.js; data driven; 3D simulation

## 0 引言

飞行器设计领域高技术、高耗费、高难度等特点,使得飞行实验及其成果复现都受到现实条件的约束,难以在实际空间中进行[1]。基于这种现状,通过可视化仿真技术将已有飞行器设计所得数据集进行三维可视化,模拟飞行行为,已经成为研究人员的必然选择。目前,绝大部分飞行仿真系统是基于 Vega Prime[2],OSG[3],Unigine[4],STK[5] 等成熟的三维引擎实现,并以传统的独立富客户端形式来展现三维仿真效果。这种方式有效地实现了飞行仿真,但由于计算资源的处理均由本地机器负责,对客户端硬件的要求较高,且资源部署不方便,不便于后期更新维护。这不可避免地使系统在应用条件和使用空间方面存在缺陷,不利于设计人员之间的协同交流,尤其是飞行器研究不再局限于设计人员内部之间交流探讨时。因此,基于仿真系统受限于应用程序和硬件设备的问题,提升系统便利性,为用户提供开放的、跨平台的,并兼具实时性和交互性的仿真系统,是当前飞行器设计的一个关键问题。

通过浏览器进行三维可视化仿真,能够有效地解决上述问题,实现资源共享。在 HTML5 以前,Web 三维可视化的实现需要借助于 Flash、Siverlight 等渲染插件,所构建的可视化系统兼容性、可移植性较差,破坏了用户体验。近年来,随着 HTML5 标准的发布,以及 WebGL 等技术的快速发展,直接推动了免插件、开放、易用、跨平台仿真系统的构建,支持用户在不限制空间、时间和操作系统平台的条件下,对仿真系统直接进行访问交互。目前,HTML5 和 WebGL 技术都已经得到各个主流浏览器的支持,如 Google Chrome、Firefox、Internet Explorer 9+等。

本文针对 Web 端飞行仿真的应用需求,采用 WebGL 第三方图形库 Three.js,构建三维飞行可视化仿真系统。该系统以气动数据为驱动,实时模拟了飞行器运动轨迹,同时兼顾天空效果、尾焰特效以及摄像机漫游[6],增加仿真效果真实性,使用户可以通过浏览器端得到直观丰富的三维仿真效果。本系统已应用于某单位的多学科智能非线性设计研究过程中,为该研究过程提供了良好的可视化支持。

## 1 WebGL 和 Three.js 技术

WebGL[7] 是一种免费的、跨平台的 3D 绘图标准,是

**收稿日期**:2019-07-30; **修回日期**:2019-08-23。
**基金项目**:四川省教育厅研究项目(18TD0021);四川省军民融合研究院开放基金(18sxb024)。
**作者简介**:冯 姣(1994-),女,四川泸州人,硕士研究生,主要从事飞行器仿真方向的研究。
**通讯作者**:刘志勤(1962-),女,四川绵阳人,教授,硕士研究生导师,主要从事高性能计算、数值模拟方向的研究。

HTML5 规范的组成部分之一。WebGL 通过 OpenGL ES2.0 标准接口进行封装，支持工程师使用 JavaScript 直接调用，从而在 HTML5 的 Canvas 标签基础上实现三维场景构建。WebGL 克服了渲染插件的弊端，允许网页利用底层的图形硬件（GPU）加速功能进行三维图形渲染，因此，提升了三维场景在 Web 端渲染性能和用户体验。

WebGL[8]提供的是底层图形接口，在编程过程中要求开发人员必须了解相关的数学和图形学知识。为了简化开发过程中的渲染细节和数据结构，构建一个对开发人员更加友好的开发环境，通常在 WebGL 的基础上进行封装，形成第三方类库，包括 Three.js，Babylon.js，PlayCanvas 等。其中，Three.js[9-10]作为一个轻量级、开源免费的开发框架，凭借出色的易用性和扩展性得到了业界的广泛应用。Three. js 不仅封装了 WebGL 原始的 API，还支持 Canvas 动画、CSS 动画以及许多实用的内置对象，可以快速实例化场景、光照、摄像机、渲染器等元素，实现基本三维场景创建，提高开发效率，降低开发成本。

## 2　系统总体设计

基于 Three.js 的飞行仿真系统主要用于飞行器飞行过程的三维动态演示，要求兼具实时性和逼真性，即在实时展示飞行过程的同时，还需生动呈现仿真环境。因此，结合气动数据和建模数据，构建飞行仿真系统，总体设计如图 1 所示。

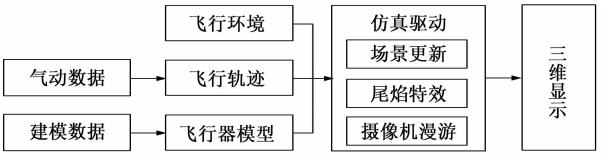


图 1　仿真系统总体设计

飞行仿真系统中的建模数据用于飞行器模型在浏览器中的重现，气动数据用于为飞行器模型的运动变化提供依据。在系统完成飞行环境、飞行轨迹以及飞行器模型的构建后，由仿真程序利用飞行轨迹驱动飞行器模型运动，并根据飞行器运动状态的不同，设置相应的尾焰特效和摄像机漫游，通过循环更新浏览器上呈现的飞行场景，使画面“动”起来，实现飞行场景在浏览器上的三维展示。

## 3　系统实现过程

### 3.1　飞行环境模拟

在仿真系统中，为了真实地对飞行状态进行可视化描述，让研究人员更好地了解整个飞行过程，飞行环境的模拟是必不可少的。飞行仿真系统的主要对象是高空释放的飞行器，其所有的飞行过程均在天空中完成，因此系统对于飞行环境的模拟主要集中在天空效果的实现方面。由于动态天空模型的计算量大、图形绘制复杂，会对系统绘制的效率和流畅性产生较大影响，所以本文采用天空盒方法[11]，简化天空效果的模拟细节。

天空效果的绘制过程如下：首先，利用 Three.js 内置对象进行场景初始化，包括场景（THREE.Scene）、摄像机（THREE.Camare）、渲染器（THREE.Render）等，由此建立起一个空白的三维仿真场景；其次，利用立方体对象（THREE.BoxGeometry）建立一个盒模型，并添加到场景中，该盒模型覆盖整个三维场景，其内部即为三维场景的可视范围；最后，选择适合的天空纹理对盒模型内部进行纹理映射，使其内部形成四周均是无限天空的视觉效果。

### 3.2　气动数据处理

气动数据集中描述了飞行器飞行状态的变化过程，是仿真系统可信度和逼真度的保证。已有的气动数据在采样过程中受现实条件限制，采样频率较低，得到的数据离散间隔过大，无法满足人眼对于流畅体验画面每秒最少刷新 24 帧要求。所以系统将离散的位置坐标进行拟合，形成曲线轨迹，丰富气动数据，为飞行器运动控制提供数据支持。

系统以气动数据为基础，利用路径类方法生成飞行轨迹，以供后续过程中改变飞行器位置使用，气动数据处理过程如图 2 所示。首先，利用 THREE. FileLoader 对象提供的 load 方法将气动数据文档加载得到数据集；然后，通过字符替换、分割等处理过程，将该数据集解析转化为便于操作的数值数组并提取三维坐标，构建飞行曲线的顶点坐标；最后，利用路径类 THREE.Path 连接各顶点，生成飞行曲线，完成飞行轨迹在三维空间中的绘制。在绘制过程中，由于 WebGL 采用右手坐标系，三维坐标转换为顶点坐标时，$y$ 值对应飞行高度，影响曲线起伏。

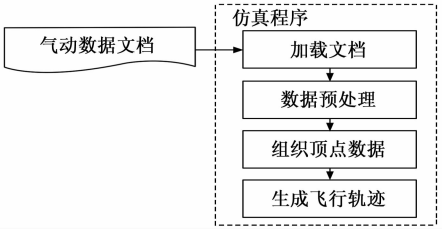


图 2　飞行轨迹生成过程

分析生成的飞行轨迹，根据飞行轨迹与水平面之间所形成的路径角 $\theta$ 的大小，可以将飞行轨迹分为三个阶段：

1）$\theta>0$，飞行器处于爬升状态，直到拉升高度达到一定值后，才逐渐将机身改平，夹角逐渐变小；

2）$\theta<0$，飞行器处于下降状态，其变化过程与爬升段相反；

3）$\theta=0$，飞行器处于爬升段和下降段之间的过渡阶段，对于巡航飞行器而言，该阶段飞行器以稳定的高度进行匀速水平飞行。

在仿真过程中，帧与帧之间的刷新间隔较短，相邻两帧间的飞行器位置连线可近似于两帧间飞行器的飞行轨迹。根据右手坐标系，如图 3 所示，设 $P_1(p_{x1},p_{y1},p_{z1})$ 和 $P_2(p_{x2},p_{y2},p_{z2})$ 分别对应相邻两帧画面中飞行器坐标，路径角 $\theta$ 求解公式如下：

$$\theta=\cot\left(\frac{|p_{y2}-p_{y1}|}{\sqrt{(p_{x2}-p_{x1})^2+(p_{z2}-p_{z1})^2}}\right) \tag{1}$$

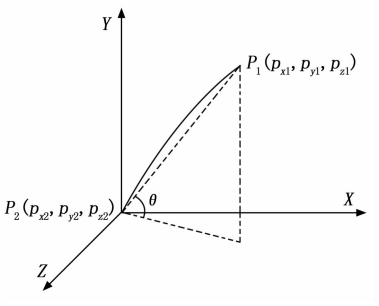


图 3 路径角求解示意图

### 3.3 飞行器模型构建

飞行器模型较为复杂，由多个几何体组合而成，采用不规则三角网的建模方法能够构建更为逼真的仿真模型，应用 Three.js 构建飞行器模型过程如图 4 所示。

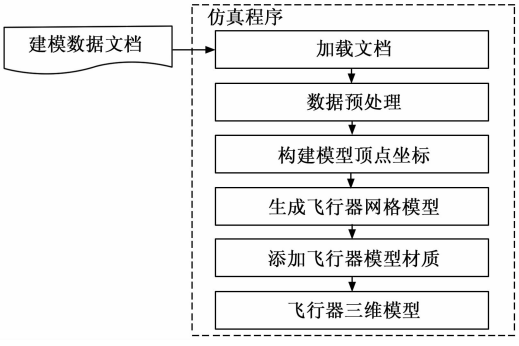


图 4 三维模型加载流程

详细过程如下：

1）建模数据预处理：向系统中传入一个 .net 格式的建模数据文件，该文件的解析过程同气动数据文档处理过程类似，通过文件加载和数据预处理，将其转化为数组格式，易于后续操作；

2）构建模型顶点坐标：利用数据中的顶点信息，结合 THREE.Vector3 方法，生成相应矢量点，并将这些矢量点保存到 vertices 数组元素中，由此组成飞行器模型的顶点坐标；

3）生成飞行器网格模型：以每个顶点在 vertices 数组中的序号作为索引，使用 THREE.Face3 生成不规则三角单元，进而生成不规则三角网，生成飞行器的网格模型；

4）添加模型材质：飞行器外形均为金属构造，因此系统选择适合金属、镜面表现物体的 Phong 材质作为模型材质，并利用 THREE.Mesh 方法，将飞行器网格模型与模型材质相结合，生成飞行器模型。

### 3.4 尾焰特效模拟

飞行器在产生动力的时候，尾部会喷射火焰。火焰的变化是一种不规则运动，具有动态性和随机性，不能精确描述，很难使用三维建模方法进行模拟。根据粒子系统方法[12]，创建粒子集，通过粒子集的动态变化来模拟尾焰的总体形态和特征，实现过程如图 5 所示。当飞行器在自身动力推动下进行飞行时，系统通过 THREE.Vector3 构造火焰粒子集，将粒子集的初始化位置设置为飞行器尾部喷口处，颜色设置为火焰颜色，并使其沿飞行器机身延展方向运动。随着飞行器和火焰粒子的运动，飞行器尾部喷口处不断产生新粒子，所有粒子沿着原来的运动方向移动。当火焰粒子与飞行器之间的距离超过模拟的火焰范围，则删除该粒子，否则将该粒子渲染到浏览器上，如此往复循环，直到飞行器进入无动力滑翔阶段。

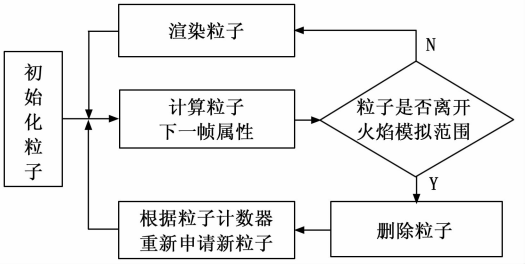


图 5 粒子系统流程

### 3.5 摄像机漫游

在各飞行阶段中，飞行器飞行特征均不相同，对于拍摄手法、场景氛围的要求存在差异。根据飞行器运动过程，对摄像机运动进行轨迹规划，合理地调整摄像机的位置和视线，使系统能够多角度地展现飞行器飞行过程，丰富场景表达，仿真效果也更为直观有效。因此，结合现代运动拍摄的知识，在气动数据处理的基础上，根据 3.2 中飞行轨迹的分段处理，为不同飞行阶段设置相应拍摄手法，以呈现不同的画面效果，摄像机运动控制如表 1 所示。

表 1 摄像机运动控制

| 飞行段 | 画面效果 | 拍摄手法 |
|---|---|---|
| $\theta>0$ | 凝视视角观察飞行器升空 | 拉镜头 |
| $\theta=0$ | 不固定观察飞行器状态变化 | 移镜头 |
| $\theta<0$ | 跟随观察目标的状态变化 | 跟镜头 |

系统对于摄像机的整体流程如图 6 所示。当飞行器位置改变时，首先调整摄像机视线，使其焦点始终对准飞行器模型，确保飞行器存在于画面之中[11]；其次，判断飞行阶段是否发生变化，如果发生变化，则根据表 1 中设定的拍摄手法的移动规则，对该飞行阶段上摄像机的移动路径进行规划，否则，判断当前状态是否满足系统设置的过渡条件，若满足，则设置相机路径为通过该点与下一飞行阶段的摄像机路径初始点之间的弧线，若不满足，则沿上一路径规划进行移动。其中，由表 1 指定的拍摄手法来调整摄像机设置的详细介绍如下：

1）$\theta>0$，飞行器逐渐飞向高空，此时固定摄像机位置，即可实现拉镜头的拍摄手法，画面效果从近景变为远景，仿真呈现仰视效果；

2）$\theta<0$，飞行器处于下降状态，为了稳定呈现飞行器运动过程，摄像机采用跟镜头，此时摄像机与飞行器保持同步运动；

3）$\theta=0$，飞行器机身处于水平状态，采用移镜头营造一个开阔的视觉效果，该拍摄手法摄像机操作自由，不受限制，系统通过选择二次贝塞尔曲线进行摄像机路径规划。

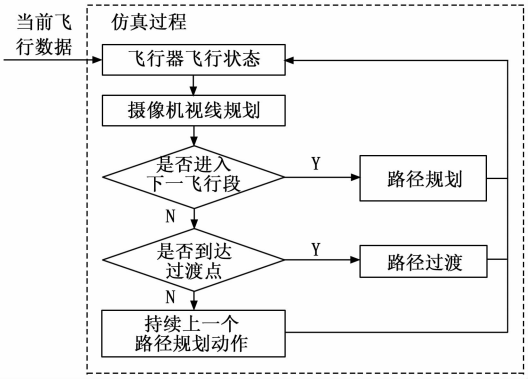


图 6　摄像机规划流程

### 3.6　场景更新

在完成飞行仿真静态场景的构造以及各个仿真要素的运动控制后，需要持续不断地进行循环渲染，才能使静态的仿真画面“动”起来。系统通过 requestAnimationFrame 循环调用仿真程序驱动场景更新。requestAnimationFrame 每调用一次，程序就更新一次飞行器在飞行轨迹中的当前位置、火焰粒子的运动状态以及摄像机视线和位置，并结合渲染器重新渲染浏览器画布内容，进行场景更新。

## 4　仿真实例

本文以匕首导弹的飞行过程为例进行飞行仿真，实现飞行过程的三维可视化。仿真的硬件平台为 PC 机，处理器是 Intel（R）Celeron（R）CPU 1005M @ 1.90 GHz 1.90 GHz，8 G 内存；软件平台是 GoogleChrome 71.0.3578.80。借助 stat.js 监控该仿真结果的 FPS 信息如图 7 所示。由此可知，除刚开始时的系统进行飞行场景建立的初始化时间外，整个仿真过程中，画面每秒传输帧数均大于 30，保证了仿真系统的稳定性和流畅性，能够满足设计人员的日常需求。

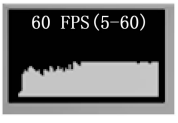


图 7　飞行仿真系统的 FPS 监测结果

图 8～10 是匕首导弹飞行轨迹的仿真结果截图。图 8 是上升段画面，可以看见匕首导弹产生动力，产生尾焰，而后在自身动力推动下开始爬升，逐渐飞向高空；图 9 飞行器处于上升段和下降段之间过渡过程，仿真画面从多个视角对飞行器状态进行展示，画面内容丰富；图 10 是下降段画面，该阶段中整体画面稳定清晰，效果良好。

图 8　爬升段仰视视角画面

## 5　结束语

针对飞行仿真系统受到应用程序和硬件设备的限制，系统开放性、兼容性、可移植性较差，无法满足用户更高

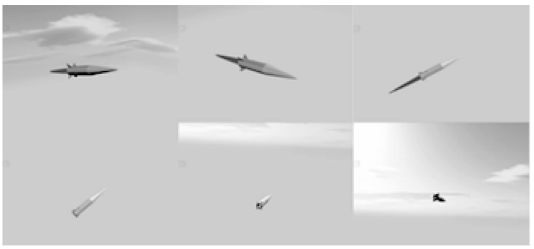

图 9　过渡段多视角画面

图 10　下降段跟随视角画面

的体验需求的实际问题，本文研究 HTML5 和 WebGL 技术，利用 Three.js 图形库，实现了基于 Web 端的飞行仿真系统。该系统模拟了飞行器模型和天空效果，通过对飞行轨迹数据进行曲线拟合，在数据的驱动下，控制飞行器模型运动、尾焰特效变化以及摄像机漫游，对飞行过程进行动态呈现。系统运行流畅，效果良好，支持用户的便捷访问，基本可以满足轨迹数据在各个操作系统平台上快速且直观的展示需求，对于三维场景在浏览器端的模拟实现，具有一定的实际实用价值。

**参考文献：**

[1] 张庆军，张明智，吴　曦. 空间作战体系建模和体系贡献度评估研究综述 [J]. 计算机仿真，2018，35（1）：8－12，17.
[2] 姜光焱，张　伟，段　昶. 基于 Vega Prime 的导弹仿真系统的研究 [J]. 系统仿真学报，2013，25（4）：645－649，667.
[3] 闫晓东. 基于 OSG 的飞行视景仿真平台开发 [J]. 计算机仿真，2008（5）：58－60，198.
[4] 王远明，卢　宽，贾　倩，等. 基于 Unigine 的舰载航空视景仿真技术研究 [J]. 系统仿真学报，2017，29（9）：2087－2092.
[5] 徐增文，师　鹏，赵育善. 基于 STK 的电磁航天器编队飞行仿真研究 [J]. 计算机仿真，2015，32（5）：95－99.
[6] 史红兵，张毅彬，童若锋，等. 虚拟场景自动漫游的路径规划算法 [J]. 计算机辅助设计与图形学学报，2006（4）：592－597.
[7] Zakas N. Javascript 高级程序设计 [M]. 曹　力，等译. 北京：人民邮电出版社，2012.
[8] 任宏康，祝若鑫，李风光，等. 基于 Three.js 的真实三维地形可视化设计与实现 [J]. 测绘与空间地理信息，2015，38（10）：51－54.
[9] 高齐琦，江　婷，田世隆，等. 基于 Three.js 的 3D 磁盘阵列设计 [J]. 计算机系统应用，2018，27（11）：241－246.
[10] Dirksen J. Learning Three. js：The 3D JavaScript 3D Library for WebGL [M]. Packt Pulishing，2013.
[11] 尹宝瑞. 基于 OpenGL 虚拟海洋环境仿真 [D]. 哈尔滨：哈尔滨工程大学，2010：47－49.
[12] 廖炎平，刘　莉，杜小菁，等. Vega Prime 实时视景仿真中粒子系统的应用研究 [J]. 系统仿真学报，2010，22（4）：938－941.