西南科技大学

Southwest University of Science and Technology

本科毕业设计（论文）

题目名称：一体化飞行器在线模拟飞行仿真系统研究

学院名称	计算机科学与技术学院
专业名称	计算机科学与技术
学生姓名	黄佳乐
学号	5120187580
指导教师	陈立伟 教授
	黄俊 讲师

二〇二二年六月

西南科技大学

本科毕业设计（论文）学术诚信声明

本人郑重声明：所呈交的毕业设计（论文），是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

作者签名：

日期：　　年　　月　　日

西南科技大学
本科毕业设计（论文）版权使用授权书

本毕业设计（论文）作者同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权西南科技大学可以将本毕业设计（论文）的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本毕业设计（论文）。

保密□，在___年解密后适用本授权书。

本论文属于

不保密□。

（请在以上方框内打“√”）

作者签名：　　指导教师签名：

日期：　　年　月　日　　日期：　　年　月　日

一体化飞行器在线模拟飞行仿真系统研究

摘要：一体化飞行器是当前各国研究热点，基于计算机模拟技术对其飞行仿真有利于观察飞行过程，且能够以较低成本辅助研究。传统 Web 三维飞行模拟一般采用 Flash、Siverlight 等插件的方式。针对插件技术兼容性差，加载效率慢等问题，以 Web 端为载体，基于 WebGL 第三方图形库 Three.js 技术，研制一体化飞行器在线模拟仿真系统。该系统无需安装任何插件，能够快速直接实现三维运动渲染。

系统以气动数据为驱动，按控制时序模拟一体化飞行器飞行轨迹、飞行姿态，并利用 Echarts 可视化控件展示各参数变化曲线；系统通过粒子模型模拟飞行器尾焰，通过多视角变化实现飞行器不同姿态的展示，以增强飞行仿真的效果。经 X-51A 虚拟飞行仿真测试，本系统能够辅助一体化飞行器轨迹设计，且能在一般计算机上获得较好性能体验，能够为飞行器设计仿真研究提供参考。

关键词：Web 三维可视化；WebGL；Three.js；Echarts；三维仿真

Design of On-line Flight Simulation System for Integrated Vehicle

Abstract: Integrated vehicle is currently a research hotspot in various countries, and its flight simulation based on computer simulation technology is conducive to observing the flight process, and can assist research at a lower cost. Traditional Web 3D flight simulation generally uses plug-ins such as Flash and Siverlight. Aiming at the problems of poor plug-in technology compatibility and slow loading efficiency, an integrated vehicle online simulation system is developed based on the WebGL third-party graphics library Three. js technology with the Web terminal as the carrier. The system does not need to install any plug-ins, and can quickly and directly realize 3D motion rendering.

The system is driven by aerodynamic data, simulates the flight trajectory and flight attitude of the integrated vehicle according to the control sequence, and uses the Echarts visual control to display the change curve of each parameter; the system simulates the tail flame of the aircraft through the particle model, and realizes the display of different attitudes of the aircraft through multi-perspective changes. , to enhance the flight simulation effect. After the X-51A virtual flight simulation test, the system can assist the trajectory design of the integrated vehicle, and can obtain better performance experience on the general computer, which can provide a reference for the simulation research of aircraft design.

Key words: Web 3D visualization; WebGL; Three.js; Echarts; 3D simulation

目　录

第 1 章 绪论

1.1 研究背景

中国制造业规模占全球近 30%，但工业软件的市场份额不足 6%，至今，国内 80%的规划软件、50%的制作软件、95%的效劳软件无法做到自主可控[1]。国内 60%以上的大公司采用德国 SAP[2](System Applications and Products)和美国 Oracle 的软件，有的则采用法国 CATIA[3]。如果过分依靠国外的工业软件，就会使整个行业的生产活动，都要按照国外的技术、技术、设计思路、管理经验来进行，从而将我们的生产企业牢牢地锁在了产业链的最底层。此外，对国外软件的过分依赖，也带来了行业的安全问题，例如 2008 年的“微软黑屏”，2011 年的“泄密门”软件 Solidworks,2013 年的“棱镜”项目，到 2020 年，哈工程公司被禁止使用 MATLAB (MATLAB)，《矩阵实验室》，这一切都成为了国际上的共识[4]。

中国是世界上最大的制造业国家，但尚未成为制造业强国，加速发展制造业软件产业是中国由制造业大国向制造业强国转型的关键。国内软件业仍面临三大挑战：第一，软体自主创新能力较弱，高端软体仍处于被国外所垄断的状况。国内软件还没有从模仿、跟风、追赶中摆脱出来。二是由于软件与制造业的深度融合不够，国内的工业软件大多是标准的、通用的，缺乏对产品的个性化和二次开发的相关经验，很难适应企业的实际业务和具体的应用场景。三是软件业规模小、市场成熟性差、企业缺乏竞争力，目前国内软件业总体实力不强，软件业主要以 OA （OA）、 CRM （CRM）等低门槛的软体为主，对于设计定制的工业软件这一块，我国还处于比较薄弱地位，面对国产软件产业现状，既是挑战也是机遇。

鉴于当前国内工业软件主要依靠国外的产业痛点，习近平主席在十二届人大三次会议上第一次把“军民融合”提到了国家战略层面。绵阳市是四川省第二大城市、中国唯一的科技城，是全国重点的军事、科研、生产基地，拥有中国工程物理研究所 18 个，以及其他 14 所高校。绵阳是四川军民融合发展的前沿阵地，肩负着国家深化改革和军民融合战略探索、积累经验、以身示范的重大使命。

以军民融合为契机，研发飞行器设计等大型专业国产软件，增强国产软件自主创新，为我国国产软件的研发提供积累经验。

本课题针对我国软件产业与制造业深度融合不足这一痛点，对一体化飞行器仿真飞行系统进行研究。一体化飞行器[5]，气动系统与推进系统相耦合，具有良好升阻特性和

进气道流量捕获特性，是新型高超声速飞行器研制的重要方向。在飞行器设计阶段，飞行器在现实空间中进行飞行实验，不仅成本高，且受到实际条件限制[6]，利用仿真技术对飞行器飞行轨迹及飞行姿态进行实时模拟，可有效减少实际飞行测试次数，减少飞行器设计研发成本。

可视化仿真技术的目的在于为用户营造一种“身临其中”的虚拟环境，随着该技术的快速发展，目前已经广泛地应用于教育、制造、员工训练等各个领域。飞行仿真是可视化仿真技术在航天航空领域的一种应用方式，它通过对抽象复杂的气动数据进行形象化，构造仿真环境，尽可能逼真地刻画仿真对象在整个仿真过程中的状态变化，为用户提供清晰直观的表现形式和观察角度，使其能够更快更准确地验证其仿真结果的正确性[7]。

1.2 国内外研究现状

从上世纪八十年代开始，美国、英国等西方国家就已经开始利用计算机模拟三维环境进行飞行仿真，且取得了众多成果。美国空军于 2019 年一月宣布，其 412 电子战部队将在爱德华兹空军基地建立一个联合模拟环境，以提供一个可扩展，高保真，通用的模拟环境[8]。2017 年 3 月，美国宇航局发布了 2017-2018 年度软件目录，其中提供了空间模拟环境，可以为飞船开发过程中的各个环节提供模拟，为飞船任务分析、运行控制和计划提供依据。美国空军在 2014 年启动了一项名为“真实-虚拟-结构-先进训练环境”的技术示范计划，旨在加速美国飞行员在实际操作中的应用[9]。图 1-1 为该系统效果图。

目前，仿真系统的主要实现技术集中于利用 Unigine[10]、OpenGL[11, 12]、Vega Prime、OSG[13]等成熟的三维引擎来构建独立富客户端的形式[14]。该形式对计算机硬件要求高，且专业性强，使用空间有限。随着互联网技术的高速发展，将 Web 技术与仿真技术结合已成为今后仿真技术研究的趋势之一[15]。

1.3 研究目的

本文研究对象为飞行器仿真飞行系统设计，以 X-51A 飞行器为例，基于 Web 技术，设计一套飞行仿真系统，该系统包括对飞行器飞行轨迹的模拟可视化，飞行器姿态以及多视角观察飞行器飞行状态，且兼顾天空效果和气动数据实时可视化。同时，系统还提供了用户交互功能，支持用户在 Web 端对飞行器飞行姿态进行截图，对飞行器视频进行下载并保存到本地，方便相关科研人员交流探讨。该系统能够在飞行器设计初期，为飞行器设计相关人员提供参考价值。

1.3.1 构建在线飞行器可视化仿真交互平台

基于 Web 技术，利用 WebGL 第三方图形库 Three.js 技术，构建飞行仿真系统，真实再现飞行器的飞行过程，初步支持研究人员快速验证飞行器控制算法设计等相关理论研究，为飞行器设计领域提供强有力的分析依据，支持跨平台，在线交互使用，提高系统的可移植性和通用性。利用 Echart 可视化控件技术，实现了对飞行器气动数据实时可视化，并实时展示各参数变化趋势，方便飞行器设计相关工作人员实时观测飞行器每一时刻飞行状态，有利于工作人员对飞行器各参数变化趋势有一个直观清晰高效的总体观察。

1.3.2 研究三维模型的建立

在飞行仿真平台中，为了对整个飞行环境进行展示，需要对天空、大地、飞行器、烟雾等多种事物进行三维模拟。按照种类，可以将三维模型简单的分为：背景模型、复杂实体模型以及粒子模型。在描述模型的过程中，所采用的多边形数量越多，则模型越加逼真，同时，用于模型渲染的时间也就越长，从而影响系统的实时性和沉浸感。所以，针对不同的模型，仿真平台应对不同的仿真要求，采取不同的建模方法。本课题中，对飞行仿真系统中的仿真要求进行具体分析，确定适应的仿真方法，对不同三维模型进行模拟实现，在不损害仿真平台实时性的基础上，力求实现仿真动画的真实感。

1.4 本章小结

本章主要介绍了中国工业软件高度依赖于国外，这种状况在某种程度上阻碍了中国从一个生产大国到一个生产强国的转型。绵阳市是中国唯一的高新技术城市，它肩负着“以民为先”的重任，以军民融合为契机，研制大型专业国产软件，提高我国自主创新能力，为我国国产工业软件的设计与研发提供和积累经验。目前飞行仿真软件主要为成本高，专业强，使用空间受限的独立客户端的形式，本课题提出了一种基于 Web 技术的仿真飞行平台。

第 2 章 仿真需求分析

2.1 功能需求

为了构建飞行器飞行仿真平台，初步支持研究人员快速验证飞行器控制算法设计等相关理论研究，为飞行器设计领域提供强有力的分析依据，支持跨平台，在线交互使用，提高系统的可移植性和通用性。利用 Echart 可视化控件技术，实现了对飞行器气动数据实时可视化，并实时展示各参数变化趋势，方便飞行器设计相关工作人员实时观测飞行器每一时刻飞行状态，有利于工作人员对飞行器各参数变化趋势有一个直观清晰高效的总体观察。飞行仿真系统功能布局如图 2-1 所示，具体功能需求如下。

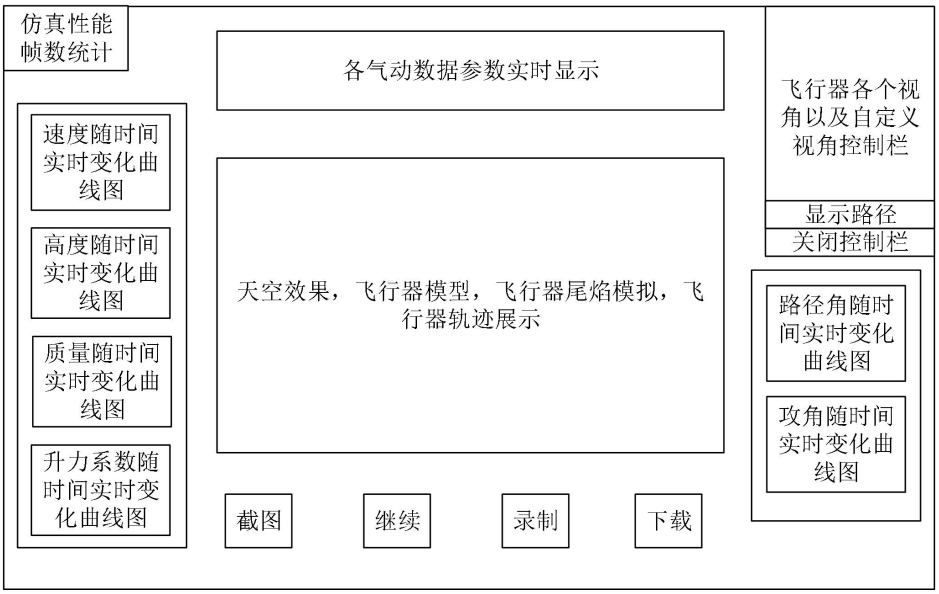

图 2-1 仿真飞行系统功能布局总览

（1）仿真性能帧数统计：系统要求对飞行器仿真飞行动画进行帧数实时统计，以检测飞行仿真系统仿真帧数是否达到人眼分辨率 24 帧。

（2）速度-时间曲线图：系统要求实时向用户展示飞行器速度与时间变化曲线图。

（3）高度-时间曲线图：系统要求实时向用户展示飞行器高度与时间变化曲线。

（4）质量-时间曲线图：系统要求实时向用户展示飞行器质量与时间变化曲线，方便用户观察飞行器燃料是否燃尽。

（5）升力系数-时间曲线图：系统要求实时向用户展示飞行器升力系数与时间变化曲线。

（6）各气动数据实时显示：气动数据与时间曲线图可以反应各气动数据在飞行器飞行阶段变化曲线，为了满足用户快速准确得到某一时刻各气动数据数值，系统要求实

时展示某一时刻气动数据数值。

（7）天空效果：为了尽可能得到与人眼看到的真实天空的仿真效果，系统要求为用户提供逼真的天空效果。

（8）飞行器模型：系统要求向用户对给定的飞行器进行模拟。

（9）飞行器尾焰：为了尽可能模拟飞行器飞行，系统要求在飞行器尾部增添火焰粒子。

（10）飞行器飞行轨迹：由给定的气动数据可以计算出飞行器的飞行轨迹，系统要求将飞行器飞行轨迹可视化展现，方便用户对轨迹数据进行研究和分析交流。

（11）截图：系统要求对飞行器飞行的某一时刻进行截图，方便用户间进行分享，交流与讨论。

（12）录制：系统要求在某一时间段内对飞行器飞行过程进行录像，方便用户间进行分享，交流与讨论。

（13）暂停与继续：系统要求为用户提供暂停与继续功能，方便用户随时暂停观察飞行器各参数随时间变化曲线和飞行器飞行姿态。

（14）飞行器各视角与自定义视角控制栏：为满足用户随意切换观察飞行器视角，系统要求设计对飞行器观察视角的控制栏，方便用户操作。

（15）显示路径：系统要求对飞行器飞行轨迹可以进行隐藏。

（16）关闭控制栏：系统要求飞行器视角控制栏可以随时关闭。

（17）路径角-时间变化曲线：系统要求向用户展示飞行器路径角随时间实时变化曲线。

（18）攻角-时间变化曲线：系统要求向用户展示飞行器攻角随时间实时变化曲线。

2.2 性能需求

（1）本系统要求能在一般计算机上运行，且运行仿真动画要求在 24 帧以上。

（2）本系统要求点击“开始”，“截图”，“录制”，“下载视频”响应时间介于 400ms~2000ms 之间。

（3）本系统要求视角控制栏所有响应时间控制在 400ms~2000ms 之间。

（4）系统能在高于实际系统运行压力 1 倍的情况下，稳定的运行 12 小时 。

2.3 本章小结

本章主要对飞行器仿真需求进行分析，系统以 X-51A 飞行器为例，研究飞行器的飞行姿态，飞行轨迹，同时兼顾尾焰效果和天空效果，在此基础上，系统向用户提供了各气动数据随时间的变化曲线。系统还向用户提供了交互功能，包括飞行器视角控制栏、截图、录屏、暂停和继续和下载录像。该仿真系统要求在一般计算机能够运行，且要求仿真动画在 24 帧以上，点击事件响应事件介于 400ms~2000ms 之间，且要求系统不间断仿真飞行时长不低于 12 小时。

第 3 章 系统架构

3.1 总体架构

该项目采用 Web 开发方法，通过分析用户在服务器端采集的飞行数据，对其进行分析，生成相应的数据，生成相应的数据，并将所需要的数据传送给客户机，从而大大减少了服务器向客户端的数据量，从而提高了系统的运行效率，降低了服务器端的负荷，客户端利用 Three.js 应用框架，调用 WebGL 三维引擎，实现飞行数据的三维仿真，其原理如图 3-1 所示。

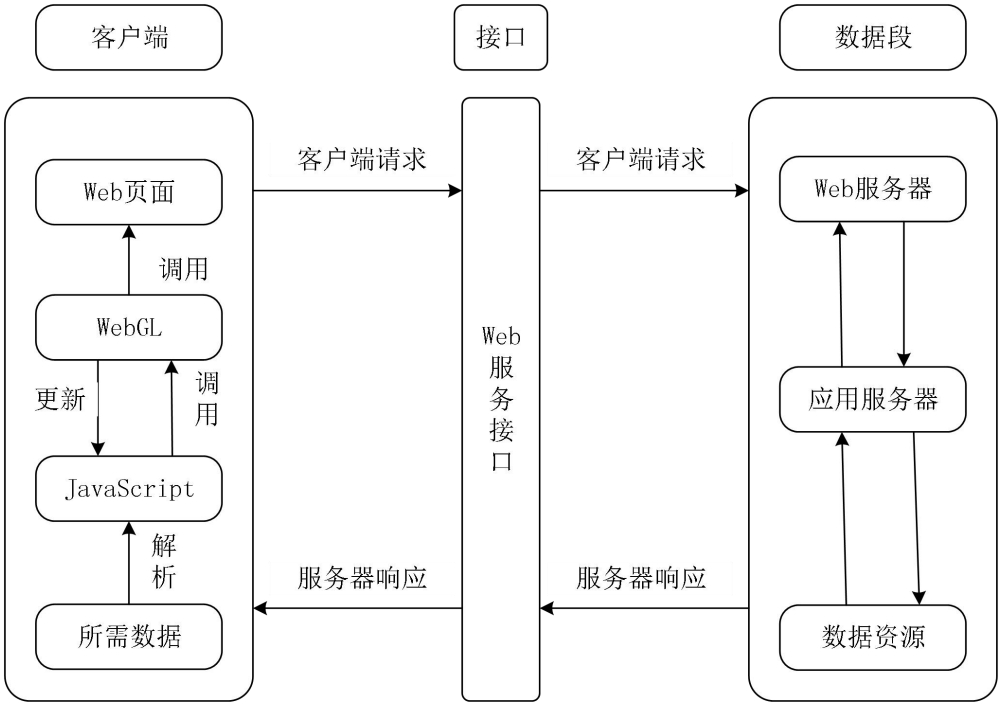

图 3-1 基于 Web 方式的飞行仿真系统原理

在该 B/S 系统结构中，其工作原理是：

（1）用户在客户端上传飞行数据，通过 Web 服务接口，提交 Web 服务器；

（2）服务器进行飞行数据的解析、提取、计算，得到仿真所需数据，包括飞行过程仿真数据，并通过 Web 服务接口响应用户请求；

（3）所需数据以 JSON 数据源的形式返回客户端，通过客户端 JavaScript 脚本解析数据；

（4）使用 Three.js 应用框架，调用 WebGL 对模型数据和矩阵数据进行渲染；

（5）最后，将渲染结果传递给 Web 页面，通知 Web 页面更新飞行数据的仿真结果。

3.2 基于 Flask 后端框架

Flask 是一种轻量级、可自定义框架，它比其它同类框架更加灵活、轻便、安全和易于上手。Flask 与 MVC 模型相结合，开发人员可以通过小团队的协作来快速的实现强大的中小型站点和 Web 服务。而且 Flask 具有很强的自定义能力，可以根据自己的需要增加功能，既可以简化核心功能，又可以扩充功能，而且它的强大插件可以让使用者进行个性化的定制，从而形成一个强大的站点[16]。

Flask 的基础结构是在一个程式中指定一个视图功能，每次使用者存取此 URL 时，系统会执行指定的视图功能，取得返回的值，并在浏览器中显示，如图 3-2 所示。

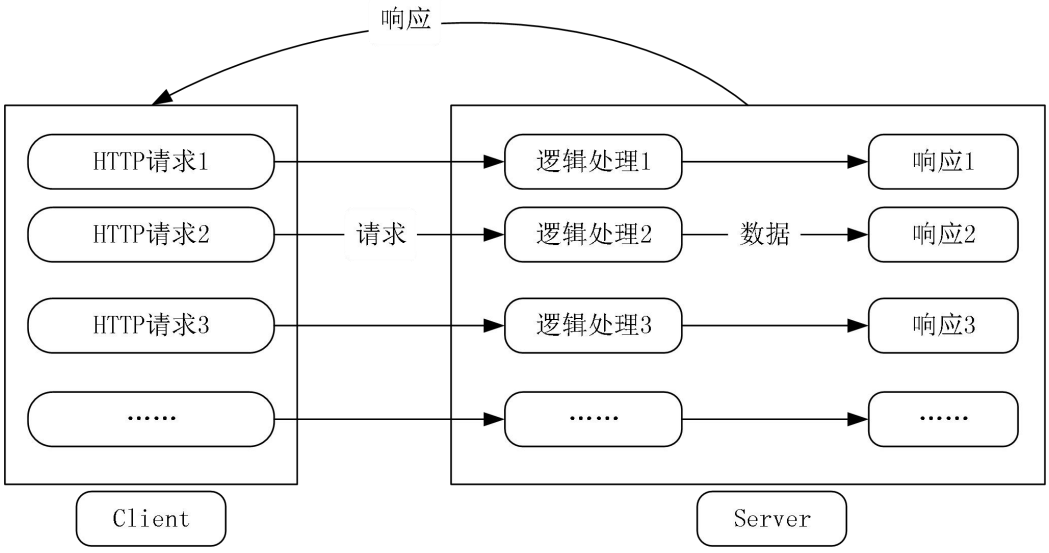

图 3-2 Flask 框架工作过程图

3.3 基于 Web 前端框架

3.3.1 HTML5，CSS，JavaScript 技术简介

HTML5 是 HyperText Markup Language 5 的缩写，HTML5 技术是将 HTML4.01 技术与现有的 HTML4.01 技术相结合，并进行了创新，适应了现代网络发展的需要。HTML5 是由多种技术组成的，它被广泛地用于因特网，为提高网络应用的性能提供了更多的标准设备。HTML5 在语法特性上比传统技术更为突出，同时也将 SVG 的内容融入其中。同时，HTML5 也将其它的元素整合到一起，对原有的功能进行调整、修改，实现规范化。到 2012 年，HTML5 已经有了一个稳定的版本[17]。

HTML5 是一种用来描述 Web 内容的语言。HTML5 是下一代因特网标准，它是用来构造和展示因特网内容的一种语言.HTML 在 1990 出现，1997 年 HTML4 作为 Internet

标准，在 Internet 上得到了广泛的应用。

CSS 提供了 HTML 标签语言的风格说明，它定义了如何显示这些元素。CSS 是 Web 设计的一次突破。使用这个功能，你可以通过修改一个小的风格来更新与之有关的网页元素。CSS 具有以下特点：

（1） 丰富的风格定义：CSS 可以为文档风格提供丰富的外观，也可以设定文字和背景属性；允许为任意元素建立边界、元素边界和其它元素之间的间距、元素边界和元素内容之间的间距；可以任意更改文字的大小写、修饰和其它网页的影响。

（2）容易使用和修改：CSS 可以在 HTML 的风格属性中定义风格，或者在 HTML 文档的头部分，或者在特定的 CSS 文件中声明风格，以便 HTML 网页参考。总的来说，CSS 样式表能够统一地存储各种风格的声明，并对其进行统一的管理。此外，您还可以对同一风格的项目进行分类、定义相同的风格，或者将特定的风格套用到 HTML 标签上，或者将 CSS 风格指定为特定的网页元素。如果要更改风格，我们只要在风格清单中找到对应的风格宣告就可以了。

（3）多页应用：CSS 样式可以被分开保存在 CSS 文档中，因此我们可以在不同的网页上使用相同的 CSS 样式。CSS 风格表理论上并不是网页档案的一部分，它可以被任意的网页档案所参考。这使得多个网页的样式能够达到统一。

（4）分层：简单地说，分层就是把相同的风格设定在某一项目上，然后再利用上次设定的属性值。比如，将相同的 CSS 样式表用于一个网站中的多个网页，同时一些网页中的一些元素希望使用其它风格，您可以为它们分别定义一个样式表格。之后的这些风格会重新编写之前的风格，在浏览器里会显示出最后的风格。

（5）网页压缩：当网页采用 HTML 来定义网页效果时，常常会要求大量或重复的表单和 font 元素，以形成不同的格式，导致网页上出现了许多 HTML 标记，导致网页的尺寸增大。另外，把风格的宣告放在 CSS 样式表中，可以极大地缩小网页的大小，从而可以极大地缩短网页的载入时间。此外，采用 CSS 样式表可以极大地减少网页的大小，降低下载速度。

JavaScript 是一种轻型、解释性或即时编译的编程语言，它的功能是优先的。尽管它以编写网页的脚本语言著称，但在许多非浏览器环境下， JavaScript 基于原型编程，多范式的动态脚本，支持面向对象，命令式，声明式，函数式编程范式。

JavaScript 脚本语言具有以下特点:

（1）脚本语言：JavaScript 是一种由 C、C++等语言编写的解释性语言。

（2）面向对象：JavaScript 是一种能够同时创建对象和已存在对象的基于对象的脚本语言。

（3）简化：JavaScript 语言中的变量类型是一种很弱的类型，它不需要对所用的数据类型进行严格的定义，它是一种以 Java 为基础的代码和控件为基础的脚本语言，它的结构非常简洁。

（4）动态：JavaScript 是一种使用事件驱动的脚本语言，能够对使用者的输入作出反应，而无需通过 Web 服务器。当访问一个页面时，JavaScript 可以通过鼠标在页面上单击、上下移动、窗口移动等方式来对这些事件做出反应。

（5）跨平台：JavaScript 脚本语言没有操作系统的依赖性，只需要一个浏览器就可以了。所以，如果你的计算机上有一个浏览器，那么 JavaScript 脚本就能在你的计算机上运行。

3.3.2 WebGL 与 Three.js 技术简介

（1）WebGL （web Graphics Library）是一种能够将 JavaScript 与 OpenGL ES2.0 相结合的 3D 绘图协议，它可以为 HTML5 Canvas 的硬件 3D 渲染，从而使 Web 开发者能够更顺畅地在浏览器中显示 3D 场景和模型，并建立复杂的导航和数据可视化。WebGL 技术规范不需要再设计专门的网页渲染插件，可以用来制作复杂的三维网页。

WebGL 的工作流程图如图 3-3 所示。在 WebGL 的绘制中，首先要获得顶点的数据，这些顶点数据包含目标顶点坐标、索引（控制三角绘制次序）、 uv （确定贴图坐标）、法线（确定照明效果）以及不同的矩阵（投射矩阵）；其次，产生一个顶点着色，它是用 Javascript 来定义一个顶点着色程序的一个字符串，然后产生并编译为一个图形程序，传送到 GPU；接着是图元组装， GPU 按照顶点数目，逐个进行顶点着色，并产生顶点最后的座标，实现了坐标变换；然后，在产生片元着色器的过程中，决定了色彩、质感、光照和阴影；最后是光栅化，利用薄片着色器，将各个单元的色彩进行分类，并根据缓存区的大小，判定哪个单元被遮挡，无需进行渲染，最后将单元信息存入色彩缓存区，从而实现整体渲染。

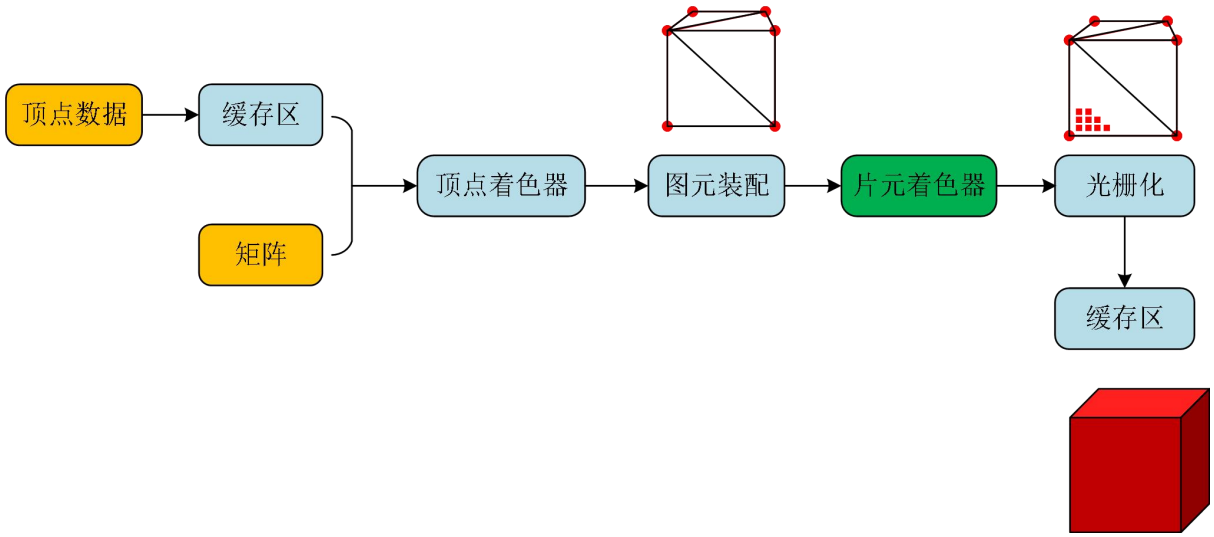

图 3-3 WebGL 的工作流程图

（2）Three.js 是一个基于 WebGL 的类库，它包含了一些工具和绘制周期，这些都是 3D 呈现要求所必需的。WebGL 的门槛比较高，而 Three. js 封装了 WebGL 所提供的界面，从而简化了许多细节，并极大地减少了学习的费用。Three.js 完成运行流程如图 3-4 所示。首先，创建场景，在创建场景的基础上增添灯光和相机；然后增添模型，对模型进行增添材质；最后对创建的场景进行着色，最终完成渲染。

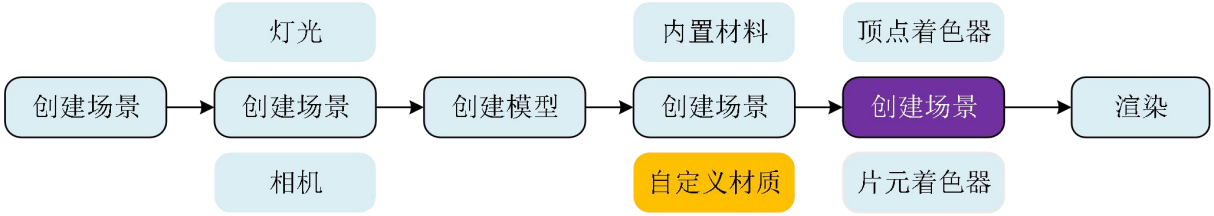

图 3-4 Three.js 完整运行流程

3.3.3 Echarts 技术简介

ECharts 是一个以 JavaScript 为基础的数据可视化图表，它能在 PC 和手机上流畅地运行，与目前大多数浏览器（IE8/9/10/11、 Chrome、 Firefox、 Safari 等），支持向量图形库 ZRender，它提供了一个直观、多互动、高个人化的数据可视化图表。本课题利用 Echarts 技术实现对气动数据实时可视化。

3.3.4 可视化模块

本课题基于 WebGL 的第三方图形库 Three.js 构建虚拟飞行可视化系统，其可视化模块如图 3-5 所示。

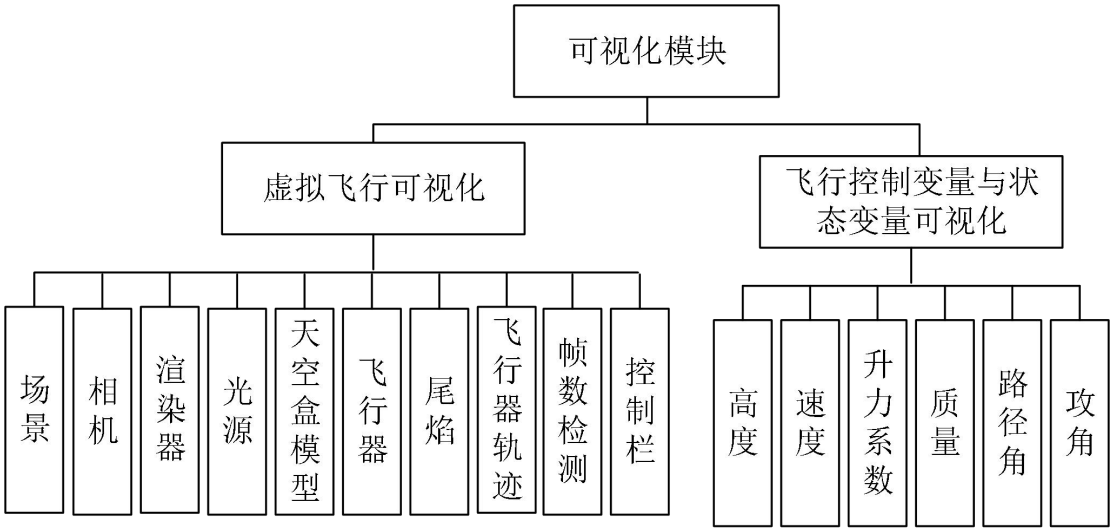

图 3-5 仿真系统可视化模块结构图

首先定义 Three.js 中三大组件，即场景、相机和渲染器；其次配置光源，构建天空盒模型[18]；然后将飞行器模型、尾焰模型及飞行器轨迹添加到场景中；然后再添加定时器与实时运动控制栏；最后为系统用户提供性能统计和控制栏。

Three.js 三大组件：场景、相机和渲染：场景是一个容器，它可以容纳所有的目标，而 Three. js 的中间场景是唯一的，它被称为 THREE. Scene，它是用一个新的对象来创建场景；照相机是摄影的一种工具，它可以通过对摄像机的方位和方向进行控制，从而获得各种角度的影像；渲染程序通过场景和摄像机来呈现，并通过使用 requestAnimationFrame 在 Javascript 中有效地持续呈现。

光源：本课题 X-51A 飞行器外壳采用的材质为光敏材质(镜面高光 Phong 材质)，这种材料要求有一个能呈现 3D 效果的光源，并且在使用时必须加入一个新的光源，不然就不会有光的效果。 Three.js 中光源的种类有点光源、平行光、聚光灯和环境光。点灯和蜡烛一样，唯一的区别就是蜡烛是有基座的，没有基座，就像是一团漂浮在半空中的火焰，从一个点发出的光是从一个点射出来的，从四面八方向四面八方扩散。平行光是模拟太阳的，而这个项目使用了平行光，所有的光都是平行的，平行光不会变弱，所以被平行光照射的区域会受到相同的强度。聚光顶的光从一个圆锥发出，在被照射的对象上形成一种聚集的效应。在传播的过程中，聚光灯也会减弱。环境光是由多个角度反射出来的光，从周围的光源中发出的光，不管它的法向量有多大，它的亮度和亮度都是一样的。

天空盒模型：在实时渲染中，当你要画很远的目标，例如远方的山脉、天空等等，

当你的视角在不断的移动时，目标的尺寸就会发生很小的改变。

Echarts 技术：其创新性地拖拽计算、数据视图、值域漫游等特点，极大地提高了用户的使用体验，使用户能够挖掘和整合数据。

飞行控制变量与状态变量实时数据可视化，采用离散数据传输，再基于 Echarts 可视控件与曲线拟合，实现飞行轨迹控制参数与状态变量，实时的、连续的变化过程可视化。

第 4 章 系统设计与实现

4.1 天空盒场景构建

在背景模型中，对于室外场景中的天空，天空圆顶或天空盒技术是普遍适用的方法。天空盒是在场景周围覆盖以矩形盒，如图 4-1 所示，加上设计好的纹理，使其生成天空的镜像。

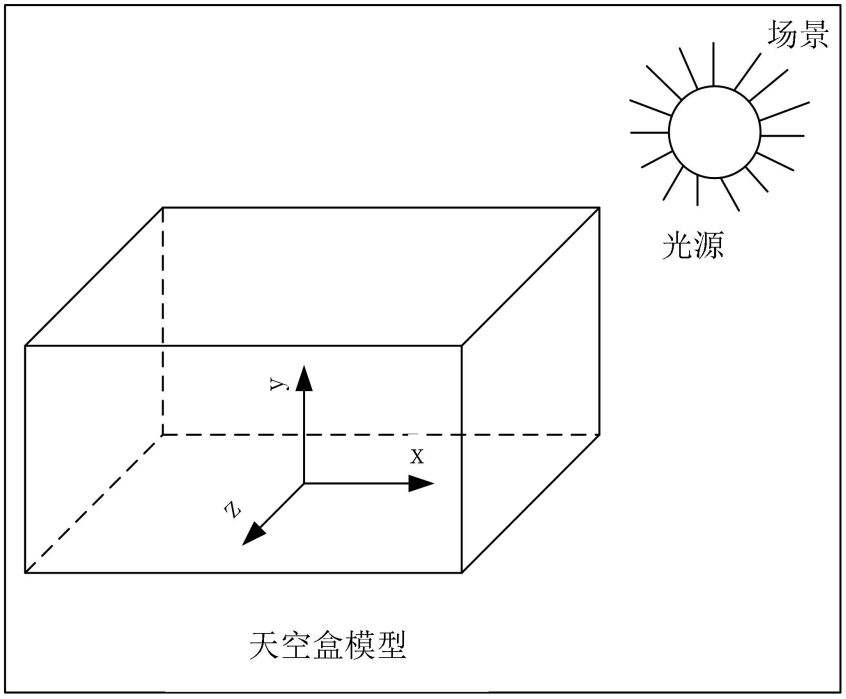

图 4-1 天空盒模型

在地形建模与模拟方面，有大量的方法，如：高程模型、基于分型的地形模拟技术、基于地形特征参数的拟合等。该项目采用了基于位图的方法来实现三维地形的快速生成，该方法主要是通过位图的灰度值与地势的高程之间建立线性映射关系，从而实现对地形的实时显示和绘制。其流程图如图 4-2 所示。具体实现步骤如下：

（1）选择适合的位图，利用公式 0.3*R+0.59*G+0.11*B 将其转换为灰度图像（R 为红色，G 为绿色，B 为蓝色）。

（2）在此基础上，提出了一种基于灰度和高度的线性映射方法。找到最大和最小的灰度图，并将其标记为 greyScaleMax 和 greyScaleMin。按实际情况设置最大高度和最小高度，分别用高度最小值表示。这样对任意灰度等级 greyScale，其高程值计算公式 4-1 为：

$$\frac{(greyScale-greyScaleMin)}{(greyScaleMax-greyScaleMin)}\times(heightMax-heightMin)+heightMin \quad (4\text{-}1)$$

式中：

greyScale---为灰度等级；

greyScaleMin---为灰度等级最小值；

greyScaleMax---为灰度等级最大值；

heightMax---为最大高程值；

heightMin---为最小高程值。

（3）计算位图中各个像素对应的高程值，并将其存入二维数组。同时，根据需要，设定沿着两维方向的步长，得到地形三维模型的所有顶点数据。

（4）计算各个顶点的法向。对四个角上的顶点，可以简单赋予其相邻面的法向；对边上的顶点，可以由两个相邻面的法向相加标准化得到；对中间顶点，可以由四个相邻面的法向相加标准化得到。

（5）利用 WebGL 的显示技术，设定光源，绘制地形。

（6）真实感场景绘制。利用纹理映射方法，将平面图像覆盖到地形表面，使地形表面形成真实感的色彩花纹。

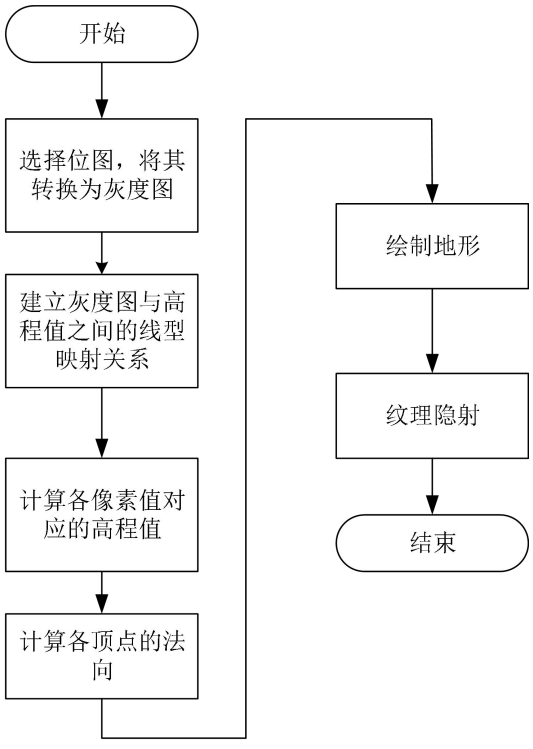

图 4-2 利用位图绘制地形算法流程图

场景是 Three.js 系统里面放置模型、光源和照相机的容器。在飞行器实时飞行渲染中，随着观察者的距离的移动，天空的大小几乎无变化，为了尽可能地模拟人眼看到真实天空的视觉效果，本文使用天空盒技术对场景的背景进行贴图，如图 4-3 所示，使之成为天空盒。图中左半部分为天空盒模型平面展开图，右部分为天空盒模型立体图。

图 4-3 天空盒模型的构建

天空盒详细设置过程如下：

（1）场景搭建：

首先，利用 new Three.Scene 构造函数创建场景对象，场景用于存放渲染的各种对象；然后，new THREE.PerspectiveCamera 构造函数用来创建透视投影相机，透视投影相机模式一般用来模拟人眼所看到的景象，为了更为接近人眼看到的场景，本课题采用透视相机，该构造函数总共有四个参数，分别是 fov，aspect，near，far 。fov 表示摄像机视锥体垂直视野角度，最小值为 0，最大值为 180，默认值为 50，实际项目中一般都定义 45，因为 45 最接近人正常睁眼角度；aspect 表示摄像机视锥体长宽比，默认长宽比为 1，即表示看到的是正方形，实际项目中使用的是屏幕的宽高比；near 表示摄像机视锥体近端面，这个值默认为 0.1；far 表示摄像机视锥体远端面，默认为 2000，本项目设置的为 1000；最后，利用 new THREE.WebGLRenderer 构造函数，创建渲染器对象。核心代码 4-1 如下。

代码 4-1 仿真场景初始化代码

```
function initWorld() {
                scene = new THREE.Scene()//创建场景对象
                camera = new THREE.PerspectiveCamera(45, window.innerWidth /
window.innerHeight, 0.1, 1000)//创建透视相机对象
                renderer = new THREE.WebGLRenderer({//创建渲染器对象
                    antialias: true, //是否执行抗锯齿
                    preserveDrawingBuffer: true, //是否保留缓直到手动清除或被覆盖
                    canvas: document.getElementById("WebGL-output")//
                })
                renderer.setPixelRatio(window.devicePixelRatio)
                renderer.setSize(window.innerWidth - 4, window.innerHeight - 4) //设置渲染器大
小
                renderer.setClearColor(0xffffff) //设置背景颜色和透明度

            }
```

（2）增添光源，为了仿真效果更加贴近于自然，本课题场景光源选择半球光源 HemisphereLight，半球光源可以使得光源可以来自天空的反射，地面的反射，以及场景中各个物体的反射。半球光源有三个参数，依次为天空发出的颜色，地面发出的颜色，以及光照强度，数值依次设置为 0xffffff, 0x554433, 1。代码 4-2 如下。

代码 4-2 场景光源初始化代码

```
    function initLight() {
                light = new THREE.HemisphereLight(0xffffff, 0x554433, 1);//天空发出光的颜色，
地面发出光的颜色，光照强度
                scene.add(light)//将光源添加到场景中

            }
```

（3）初始化天空盒模型，利用 new THREE.CubeTextureLoader 构造函数对场景 Sence 进行贴图，使之成为天空盒，使用.setPath 方法里面的字符串代表图片所在的文件夹，load 方法中的字符串为对应的六张要贴的图片，代码 4-3 如下。

代码 4-3 天空盒模型加载代码

```
    function initSkyBox() { //盒子模型
                var cubeTextureLoader = new THREE.CubeTextureLoader()
                cubeTextureLoader.setPath('img/skybox/')//读取图片文件
                skybox = cubeTextureLoader.load([
                    'px.jpg', 'nx.jpg',
                    'py.jpg', 'ny.jpg',
```

```
            'pz.jpg', 'nz.jpg'
        ])//加载图片
        scene.background = skybox //将模型赋值给场景

    }
```

初始化场景，增添光源，增添天空盒模型后，运行效果在网页端展示如图 4-4 所示，从运行效果结果可以看出，该天空效果逼真，仿真天空颜色逼近于真实天空，满足人眼观察习惯，符合天空仿真要求。

图 4-4 天空盒模型效果展示

4.2 虚拟飞行可视化

4.2.1 飞行数据定义与数据协议

模型数据：X-51A 气动外形相似的气动推进一体化飞行器型是一体化飞行器研究的典型对象，如图 4-5 所示，模型参数见文献[19]。本文基于 CAD(Computer Aided Design)软件建立模型，并导出 STL 格式数据，作为本文一体化飞行器在线模拟飞行的研究对象。

图 4-5 X-51A 模型

轨迹数据：根据美国空军研究实验室研究报告[20]中给出 X-51A 的设计轨迹信息，

如图 4-6 所示，图中为 X-51A 飞行器从开始点火起飞（Take-off）到下降段（End Flight）的飞行轨迹，本课题只针对飞行轨迹中的爬升段①，巡航段②，和无动力滑翔段③进行仿真飞行模拟。利用前期研究的自适应伪谱法[21]，计算获得轨迹的数据。轨迹数据为时序的飞行控制变量与状态变量，包含了时间(time)、飞行距离(range)、高度(attitude)、路径角(path angle)、质量(mass)、攻角(attack angle)。为实现计算机仿真与交互，计算数据进行了离散化，以 5 秒产生一条飞行控制变量与状态变量。由于完整轨迹数据量较大，论文为了便于论述数据结构，部分数据如表 4-1 所示。

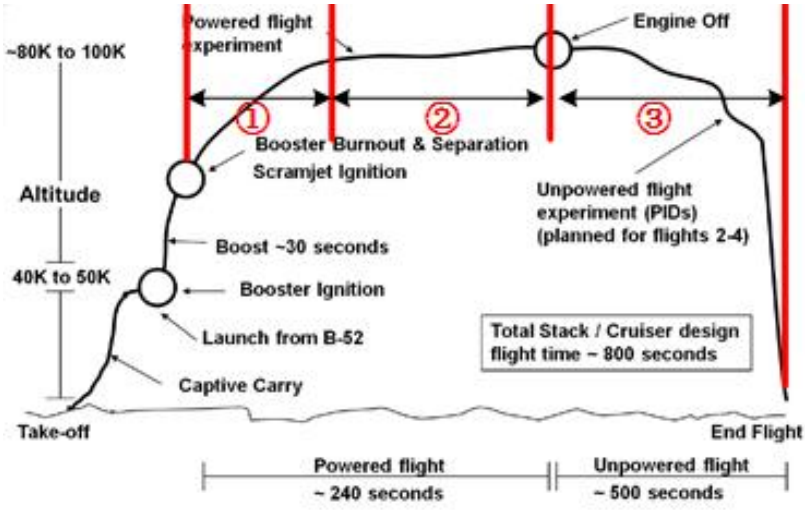

图 4-6 X-51A 飞行器设计轨迹

表 4-1 通过伪谱法计算所得 X-51A 轨迹数据与结构

Time/s	Range/m	Velocity(m/s)	Attitude/m	Pathangle/rad	Mass/kg	Attackangle(°)
0	0	1327.7	18300.0	0.1910	670.0	3.3106
5	7292.7	1346.8	19679.1	0.1822	664.8	3.4298
10	14625.5	1361.6	20985.8	0.1699	660.5	3.5620
15	22047.1	1373.1	22203.1	0.1548	656.8	3.7013
20	27037.5	1379.6	22953.1	0.1433	654.7	3.8035
……	……	……	……	……	……	……
399.8	617651.4	1008.5	978.5	-1.0443	549.9	2.0000

4.2.2 模型加载

本课题基于 CAD 建模软件对 X-51A 外型进行建模，并导出 STL 格式的飞行器模型数据，其加载流程如图 4-7 所示。首先，生成飞行器网格模型，向系统传入一个.STL 格式的建模数据文件，利用 Three.js 中加载 STL 格式的 STLLoader.js 库，创建 loader 对象，调用 load 函数构建飞行器网格模型；其次，增添飞行器模型材质，飞行器表面为金属外壳，因此系统选择镜面高光的 Phong 材质；飞行器外形颜色选择与天空颜色相称的灰亮色。在利用 Three.js 技术对飞行 THREE.Mesh 器进行仿真渲染的过程中，几何体不能直接被渲染，因此利用方法将几何体与材质结合，以实现飞行器模型的渲染。

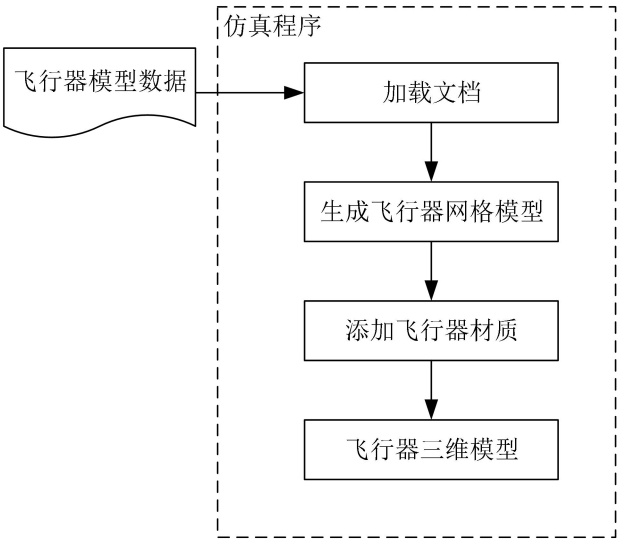

图 4-7 X-51A 模型加载流程

代码 4-4 X-51A 模型加载代码

```
function initaerocraft() {
                    var loader = new THREE.STLLoader();//
                    var mat;
                    loader.load("./x51a.stl", function(geometry) {
                        mat = new THREE.MeshLambertMaterial({
                            color: 0xdcdcdc,
                            side: THREE.DoubleSide
                        }); //0x00ffff
                        aerocraft = new THREE.Mesh(geometry, mat)
                        aerocraft.name = 'aerocraft'
                        aerocraft.geometry.center()
                        aerocraft.scale.set(0.28, 0.28, 0.28); //缩放
                        aerocraft.receiveShadow = true
                        aerocraft.castShadow = true
```

```
                scene.add(aerocraft)
                getMesh(aerocraft)
                initFire()
                updateMissle()
                updateCamera()
            });
        }
```

在飞行仿真场景创建的基础上，通过加载 X-51A 飞行器模型，将飞行器模型通过 Sence.add 方法添加到场景中，代码运行效果图如图 4-8 所示。

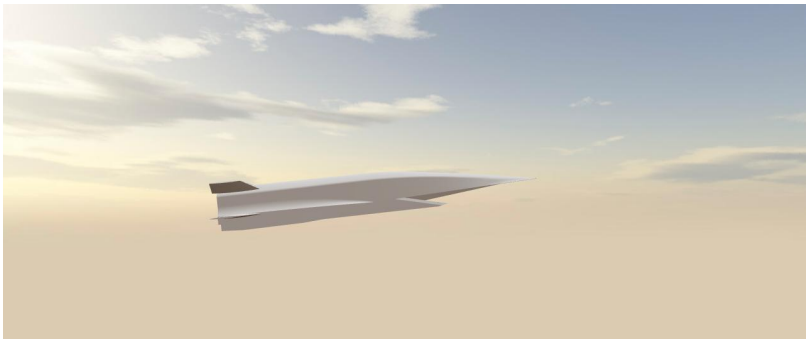

图 4-8 加载飞行器效果图

4.2.3 飞行姿态模拟

飞行器飞行姿态可由俯仰角θ 的大小来确定，如图 4-9 所示，俯仰角由速度与机体体轴的攻角α 和水平线与速度的路径角γ 共同确定。

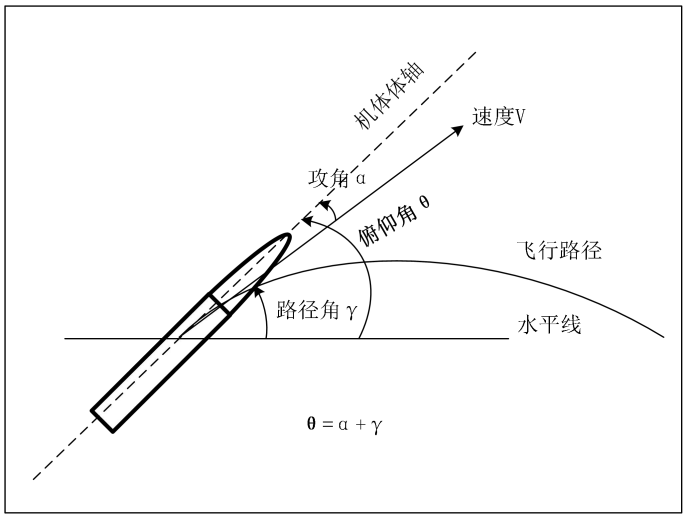

图 4-9 俯仰角示意图

如图 4-9 所示，飞行器的姿态俯仰角计算公式 4-2 为：

$$\theta = \alpha + \gamma \tag{4-2}$$

式中θ为俯仰角，α为攻角，γ为路径角

（1）$\theta>0$，飞行器处于上升状态，当上升到一定高度，路径角逐渐减小，飞行器逐渐与地面平行；

（2）$\theta<0$，飞行器处于下降阶段，路径角绝对值逐渐增大，飞行器逐渐与地面垂直；

（3）$\theta=0$，飞行器轨迹与地平面平行，且飞行器处于稳定高度的匀速飞行状态。

代码 4-5 飞行器姿态更新代码

```
function updateMissle() {
        aerocraft.position.set(pathData[T].L, pathData[T].H, 0)
        aerocraft.rotation.z = (pathData[T].pathAngle + pathData[T].attackAngle) / 180 * Math.PI
//将角度转换成弧度，rotation.z的大小表示俯仰角的大小，由路径角（pathAngle和attackAngle共同确定）
            updateFire()
            if(T%3==0){
                updateChart()
                }
            T++
            if(T == pathData.length) {
                clearChart()
                T = begin - 1
            }
        }
```

4.2.4 轨迹可视化

飞行轨迹数据加载流程如图 4-10 所示。飞行器气动数据为离散时序数据集，并以文本文件格式存储。首先，将数据集加载到系统中；然后，用字符替换，数据分割等数据预处理方法，将数据集转换成线性一维数组，并提取三维坐标；最后，将三维坐标连接起来，生成飞行器轨迹。

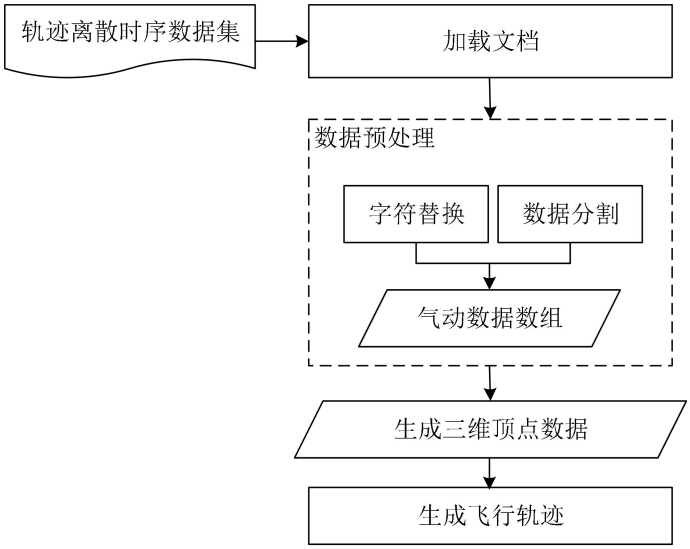

图 4-10 飞行器轨迹加载流程

气动数据存储于 PathOfAerocraft.txt 文件中，利用 new THREE.FileLoader 构造函数对 PathOfAerocraft.txt 文件进行加载，再利用字符替换等方法将文本数据存储到数组中，然后利用 THREE.Path 对象将离散的数据连接成线，通过添加飞行路径颜色和材质，最后利用 Sence.add 方法将轨迹添加到场景中。飞行器轨迹数据可视化代码实现如下。

代码 4-6 飞行器轨迹数据可视化代码

```
function initPath() {
        var loader = new THREE.FileLoader()
        loader.load('PathOfAerocraft.txt', function(data) {//加载数据
            var stringPath = data.replace(/[\r\n]/g, ' ')//字符替换
            var newString = stringPath.split('    ')//数据分割
            var trackData = []//将数据存储到trackData数组中
            for(var i = 0; i < newString.length; i++) {//遍历数组
                trackData.push(newString[i].split('\t'))
            }
            for(var i = begin; i < trackData.length; i++) {
                var pointData = new trajectory(trackData[i])
                pathData.push(pointData)
            }
            path = new THREE.Path()    //将点生成轨迹
            path.moveTo(pathData[begin - 1].L, pathData[begin - 1].H)
            for(var i = begin; i < pathData.length - 1; i++) {
                path.lineTo(pathData[i].L, pathData[i].H)
            }
            //给轨迹添加材质和颜色
            var geometry = new THREE.BufferGeometry().setFromPoints(path.getPoints())
```

```
                var material = new THREE.LineBasicMaterial({
                    color: 0xacffff
                })
                //将轨迹材质和颜色合并
                line = new THREE.Line(geometry, material)
                //将轨迹数据添加到场景中
                scene.add(line)
            })
        }
```

在飞行器模型加载的基础之上，增添轨迹数据，其展示效果如图 4-11 所示。从图中可以清楚的看到，有一条蓝色的轨迹，蓝色轨迹为飞行器轨迹可视化结果。

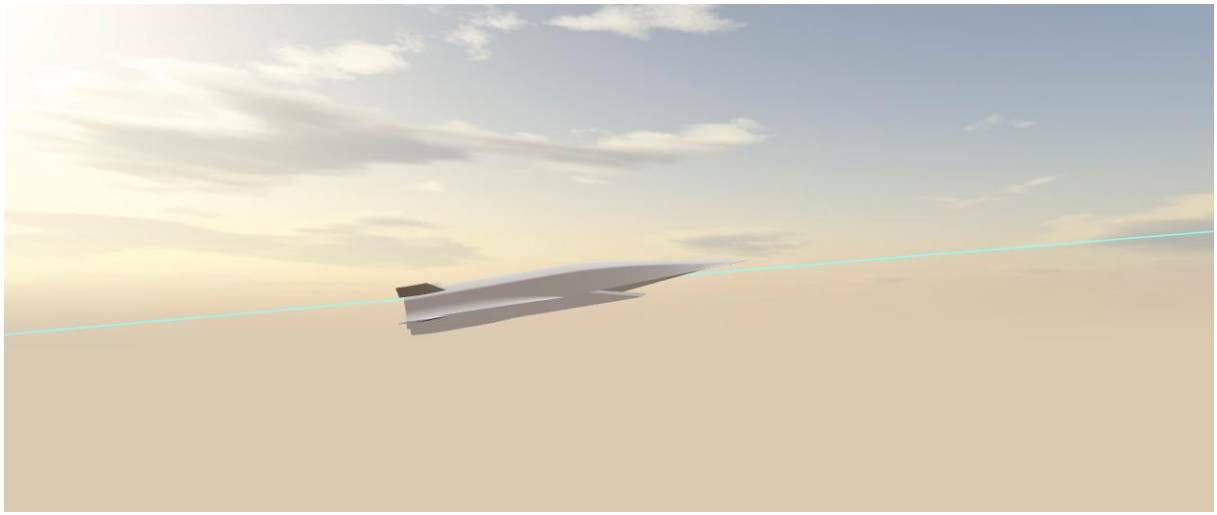

图 4-11 增添飞行轨迹后效果展示图

4.2.5 火焰粒子可视化

粒子系统是用于不规则模糊物体建模及图像生成的一种方法，其基本思想是把模糊物体定义为由成千上万个运动的、不规则的、随机分布的粒子组成的粒子集。每个粒子均有一定的生存期及其他属(如颜色、形状、大小、速度等)。粒子在不断运动的过程中改变形状，从而表现出景物的总体形态和特征的动态变化。粒子系统充分体现了模糊物体的动态性和随机性，可以很好地模拟出雨、雪、云、尾焰、烟雾等模糊对象。飞行器在飞行过程中尾部产生火焰，其具有随机性和运动性，不能精确描述。模拟不规则模糊的物体，通常有体过程法、分形法、粒子系统法[22]等，考虑到系统中的实时性和沉浸感，火焰的模拟采用粒子系统法进行尾焰效果的实现。其过程如图 4-12。飞行器燃料在充分燃烧阶段，首先，将火焰粒子位置初始化为飞行器尾部，且将粒子初始运动轨迹设置成与飞行器机身相反方向；然后，不断更新帧动画，使得飞行器尾部不断产生新的火焰粒

子，粒子的位置和运动轨迹和初始化时保持一致，当火焰粒子超出指定渲染范围时，删除该粒子，否则继续渲染该粒子，直到飞行器燃料燃尽。

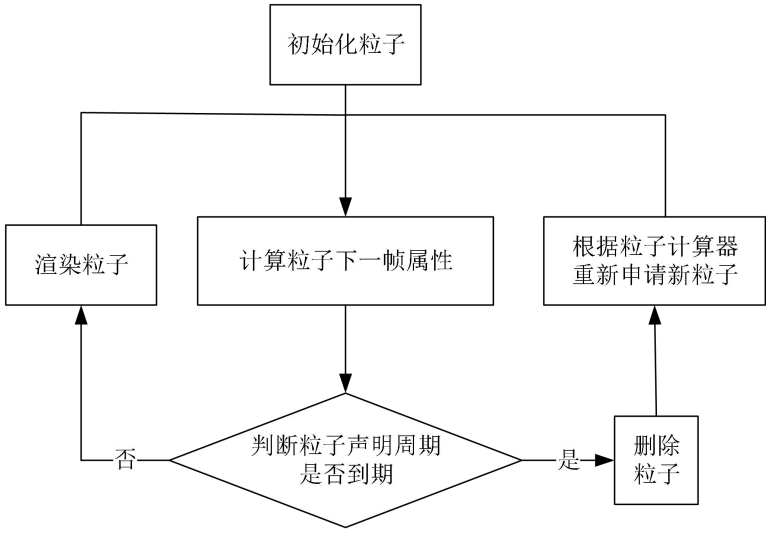

图 4-12 粒子系统流程图

粒子系统实现核心代码 5-7 如下：

代码 5-7 初始化粒子系统代码

```
function initFire() {0.

                var material = new THREE.PointsMaterial({
                    size: 0.1, //粒子大小
                    color: 0xFF4500, //粒子颜色
                    transparent: true,
                    opacity: 0.6
                })
                var geometry = new THREE.Geometry()
                var range = 3.5,
                    total = 200000 //设置粒子总数
                for(var i = 0; i < total; i++) { //对粒子状态进行依次遍历
                    var point = new THREE.Vector3(20 * Math.random(), range * (2 *
Math.random() - 1), range * (2 * Math.random() - 1))
                    if((Math.pow(point.y, 2) + Math.pow(point.z, 2)) <= 20 - point.x) {
                        geometry.vertices.push(point)
                    }
                }
                cloudPoint = new THREE.Points(geometry, material)//粒子材质
                cloudPoint.rotation.z = aerocraft.rotation.z + Math.PI//粒子在z方向位置
                cloudPoint.position.set(aerocraft.position.x, aerocraft.position.y, 0)//粒子位置
```

```
        spark = new THREE.PointLight(0xdc0000, 0.5, 1000)
        spark.castShadow = true
        spark.position.set(aerocraft.position.x - 35, aerocraft.position.y, 0)
    }
```

为了得到动态的火焰粒子系统，将粒子系统中的粒子属性进行判断，当判断到渲染粒子超过渲染的范围时，则删除粒子，当没有超过指定的渲染范围时，继续渲染该粒子。核心代码 5-8 代码如下。

代码 5-8 更新火焰粒子系统代码

```
function updateFire() {
        cloudPoint.rotation.z = aerocraft.rotation.z + Math.PI
        cloudPoint.position.set(aerocraft.position.x, aerocraft.position.y, 0)
        spark.position.set(aerocraft.position.x - 35, aerocraft.position.y, 0)
        spark.rotation.z = cloudPoint.rotation.z
        spark.intensity = bright * Math.random()
        var vertices = cloudPoint.geometry.vertices
        vertices.forEach(function(e) {
            var life = 5 * (e.z * e.z + e.y * e.y) - firelong
            e.x += 30 * Math.random()
            if(e.x > -life) {
                e.x = 0
            }
        })
        if(firelong < 80) firelong++
            //燃料耗尽 火焰变小
            if(pathData[T].Mass <= 600) {
                if(firelong > 20) {
                    firelong -= 3
                    bright = firelong / 10
                }
            }
        //动力下降 失去升力 火焰极速变小
        if(pathData[T].Lift < 0) {
            if(firelong >= 20) firelong -= 3
            if(pathData[T - 2].Lift < 0) {
                if(pathData[T - 4].Lift > 0)
                    if(params.y < 150) params.y++ //模拟失重视觉感
            } else {
                if(pathData[T + 2].Lift > 0)
                    if(params.y > -150) params.y-- //模拟失重结束视觉感
```

```
                }
            } else {
                if(firelong <= 69) firelong += 3
            }
            //熄火&重新点火
            if(firelong < 20) scene.remove(cloudPoint, spark)
            if(T == 0) {
                scene.add(cloudPoint, spark)
                firelong = 50
                params.y = 20
            }
            cloudPoint.geometry.verticesNeedUpdate = true
        }
```

图 4-13 为添加了火焰粒子模型之后的效果截图，从图中可以看出，尾焰从飞行器尾部喷出，颜色为火红色。

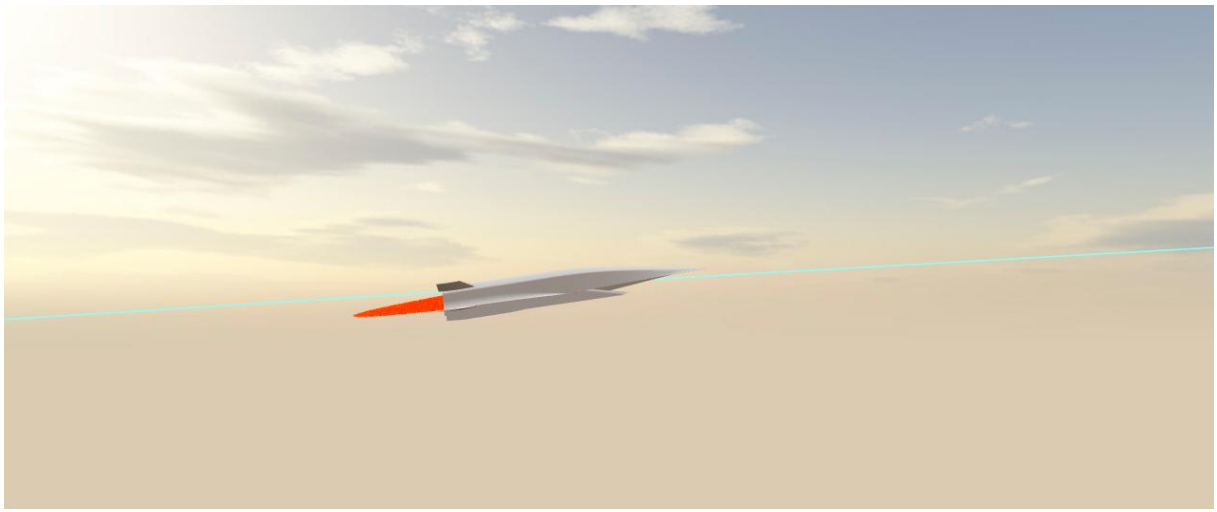

图 4-13 增添尾焰效果展示

4.3 飞行控制变量与状态变量可视化

实时性是飞行器实时飞行仿真系统的基本性能，也是保证仿真结果高度可信的重要前提，仿真系统的实时性性能直接关系到仿真结论的可信性。为了实现仿真飞行的可信性，本文首先获取浏览器端帧与帧时间间隔，如公式 4-3 所示。

$$\Delta T_i = time(render) \tag{4-3}$$

式中：

ΔT_i---为渲染动画第i帧与$i-1$帧之间时间间隔；

$time(render)$---为用于获取渲染动画帧与帧之间时间间隔；

其次根据表 4-1 中数据，基于 Lagrange 插值建立飞行器状态与时间的函数，如公式 4-3。

$$X = f(T)$$
$$T = \sum_{i=1}^{k} \Delta T_i \tag{4-3}$$

式中：

X ---为飞行器状态向量；

f ---为 Lagrange 插值函数；

T ---表示当前时间，由渲染时间求和获得；

k ---为当前渲染帧序号。

状态向量，公式 4-4。

$$X = [L, H, M_{\infty}, \gamma, M, \alpha] \tag{4-4}$$

式中：

L ---为飞行距离；

H ---为飞行高度；

M_{∞} ---为马赫数；

γ ---为路径角；

M ---为飞行器质量；

α ---为攻角。

最后通过更新帧动画的时间以达到飞行器随模拟时间运动飞行的仿真效果。

为直观高效全方位观察飞行器飞行姿势和状态，Three.js 在渲染过程中采用相机坐标系，该坐标系是在三维空间中以人眼的视点来渲染看到的场景。在相机坐标系中相机一直处于坐标原点，通过相机坐标系中 X 坐标，Y 坐标，Z 坐标来确定观察飞行器视角。

为满足直观高效的观察各气动数据随时间变化趋势，利用 Echarts 技术将飞行器各气动数据参数随时间变化曲线以图表的形式展示，该图表形式不仅可以观察到某一时刻各气动数据数值情况，而且观测到某一时间段各气动数据变化趋势，为相关工作人员提供了飞行器气动数据高效直观的展现方式，有利于相关工作人员交流和讨论。

代码 4-9 利用 Echats 技术实时展示气动数据变化代码

```
function drawChart(myChart, data, title, unit) {
                option = {
                    tooltip: {
                        trigger: 'axis',
                        formatter: '{c} ' + unit,
                        axisPointer: {
                            type: 'line',
                            animation: false,
                            label: {
                                show: true,
                                formatter: '{value} s'
                            },
                            link: [{
                                yAxisIndex: 'all',
                                xAxisName: '时间/s'
                            }]
                        }
                    },
                    grid: {
                        x: 50,
                        y: 20,
                        x2: 50,
                        y2: 20
                    },
                    xAxis: {
                        type: 'value',
                        name: '时间/s',
                        nameGap: 5,
                        data: clock,
                        boundaryGap: [0, '10%'],
                        animation: false,
                        axisLabel: {
                            interval: 99
                        },
                        axisTick: {
                            show: false
                        },
                        splitLine: {
                            show: false
```

```
                    }
                },
                yAxis: {
                    type: 'value',
                    name: title + '(' + unit + ')',
                    nameGap: 5,
                    boundaryGap: [0, '10%'],
                    animation: false,
                    axisTick: {
                        show: false
                    },
                    splitLine: {
                        show: false
                    }
                },
                series: [{
                    type: 'line',
                    data: data,
                    color: 0x99FFFF,
                    smooth: true,
                    showSymbol: false,
                    hoverAnimation: false
                }],
                data: dat
            };

            myChart.setOption(option, true);

            if(option && typeof option === "object") {
                myChart.setOption(option, true);
            }
        }
```

代码 4-10 Echarts 图表位置代码

```
<div style="position: absolute; margin:0 auto; left:25%; top: 10%; padding: 0; width: 60%;">
        <div class="info" id="info1">高度：<br />速度：</div>
        <div class="info" id="info2">升力系数：<br />质量：</div>
        <div class="info" id="info3">路径角：<br />攻角：</div>
    </div>
    <div id="chart-box1" style="position: absolute; margin: 0 auto; left: 0%; top: 8%; padding:
0; height: 85%; width: 280px;">
```

```
            <div class=" data-chart" id="chart-V"></div>
            <div class="data-chart" id="chart-H"></div>
            <div class="data-chart" id="chart-Mass"></div>
            <div class="data-chart" id="chart-Lift"></div>
        </div>
        <div id="chart-box2" style="position: absolute; margin: 0 auto; right: 5%; top: 50%;
padding: 0; height: 45%; width: 280px;">
            <div class="data-chart2" id="chart-pathAngle"></div>
            <div class="data-chart2" id="chart-attackAngle"></div>
        </div>
//气动数据实时显示函数
function updateInfo() {
            document.getElementById("info1").innerHTML = "高度：" +
pathData[T].H.toFixed(7) + " m<br/>速度：" + pathData[T].V + " m/s"
            document.getElementById("info2").innerHTML = "升力：" + pathData[T].Lift + "
<br/>质量：" + pathData[T].Mass + " kg"
            document.getElementById("info3").innerHTML = "路径角：" +
pathData[T].pathAngle.toFixed(9) + "°<br/>攻角：" + pathData[T].attackAngle.toFixed(9) + "°"
        }
```

气动数据实时可视化效果展示如图 4-14 所示，从图中可以看出，页面顶部一共有 6 个参数，分别为高度，升力，路径角，速度，质量，攻角，参数后面为某一时刻对应的气动数据数值。页面中展示处 6 个气动数据随时间变化曲线，依次为速度时间曲线，高度时间曲线，质量时间曲线，升力系数时间曲线，路径角时间曲线，攻角时间曲线，从高度时间曲线可以看出，飞行器正处于巡航段。

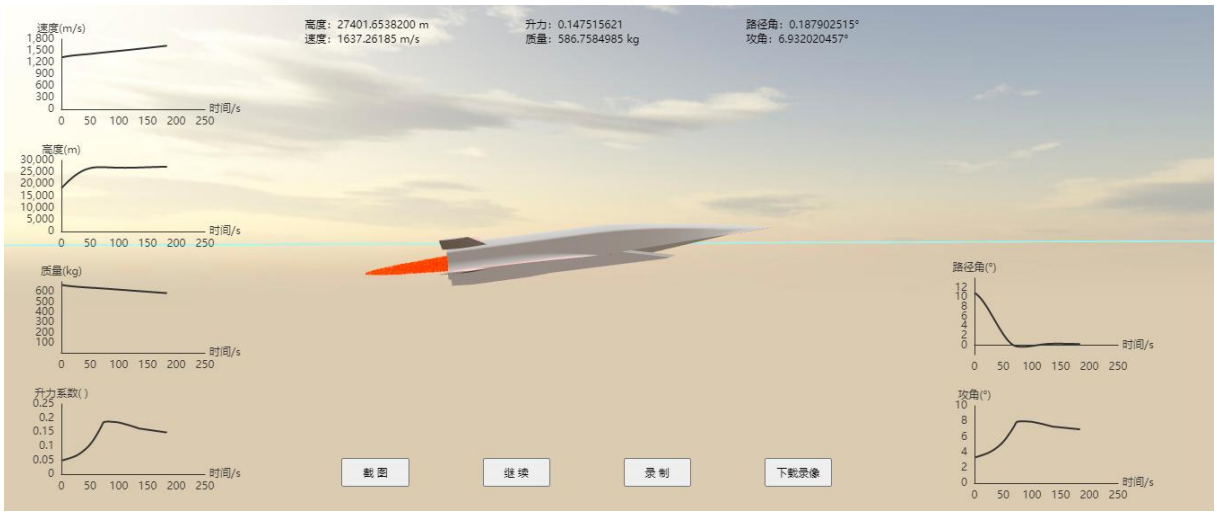

图 4-14 气动数据实时模拟效果图

4.4 用户交互功能实现

4.4.1 飞行器视角控制栏实现

首先利用 Three.js 中 GUI 组件声明一个对象，用于获取交互 UI 的 get 和 set 方法回调，初始化相机初始坐标(x:-100,y:20,z:50)，并用欧几里得范数 Math.hypot(50,20,100)函数计算出初始化观察距离，然后利用 folder.add()函数增添 x 轴，y 轴，z 轴取值范围框；其次再设计默认视角(x:-100,y:20,z:50)，背光视角(x:70,y:-50,z:70)，追光视角(x:-100,y:30,z:0)，侧方视角(x:-0,y:0,z:200)，下方视角(x:0,y:-450,z:0)，最后添加“显示路径线”和“Close Controls”。

代码 4-11 飞行器控制栏代码

```
function initGUI() {
                    params = {
                        x: -100,
                        y: 20,
                        z: 50,
                        distance: Math.hypot(50, 20, 100) //欧几里得范数
                    }
                    gui = new dat.GUI(); //创建GUI对象
                    var folder = gui.addFolder('相机视角'); //添加文件夹
                    gui.domElement.style = 'position:absolute;top:0px;right:0px'; //设置交互界面位置
                    folder.add(params, 'x', -300, 300).name('x坐标').listen()
                    folder.add(params, 'y', -150, 150).name('y坐标').listen()
                    folder.add(params, 'z', -300, 300).name('z坐标').listen()
                    folder.add(params, 'distance', 100, 450).step(1).name('观察距离
').listen().onChange(function() {
                        var scale = params.distance / Math.hypot(params.x, params.y, params.z)
                        params.x *= scale
                        params.y *= scale
                        params.z *= scale
                    })
                    folder.open()
                    params.Default = function() {
                        params.x = -100
                        params.y = 20
                        params.z = 50
                        params.distance = Math.hypot(100, 20, 50)
                    };
                    gui.add(params, 'Default').name('默认视角')
```

```
        params.Sunsight = function() {
            params.x = 70
            params.y = -50
            params.z = 70
            params.distance = Math.hypot(70, -50, 70)
        };
        gui.add(params, 'Sunsight').name('背光视角');
        scene.add( axesHelper );
        params.Track = function() {
            params.x = -100
            params.y = 30
            params.z = 0
            params.distance = Math.hypot(-100, 30, 0)
        };
        gui.add(params, 'Track').name('追踪视角');
        params.Side = function() {
            params.x = 0
            params.y = 0
            params.z = 200
            params.distance = Math.hypot(200)
        };
        gui.add(params, 'Side').name('侧方视角');

        params.Beside = function() {
            params.x = 0
            params.y = -450
            params.z = 0
            params.distance = Math.hypot(-450)
        };
        gui.add(params, 'Beside').name('下方视角');
        Visible = {
            pathVisible: true
        }
        gui.add(Visible, 'pathVisible').name('显示路径线').onChange(function() {
            line.material.visible = Visible.pathVisible
        })
        gui.close()
    }
```

实现结果如图 4-15 所示，页面右上角为飞行器视角控制栏，用户可以通过移动鼠标改变 x，y，z 坐标数值或者在其右侧小方框中键盘输入数值自定义观察飞行器视角。

为了方便用户从不同视角观察飞行器飞行姿态和状态，增添了默认视角，背光视角，侧方视角，下方视角以及是否显示路径功能。

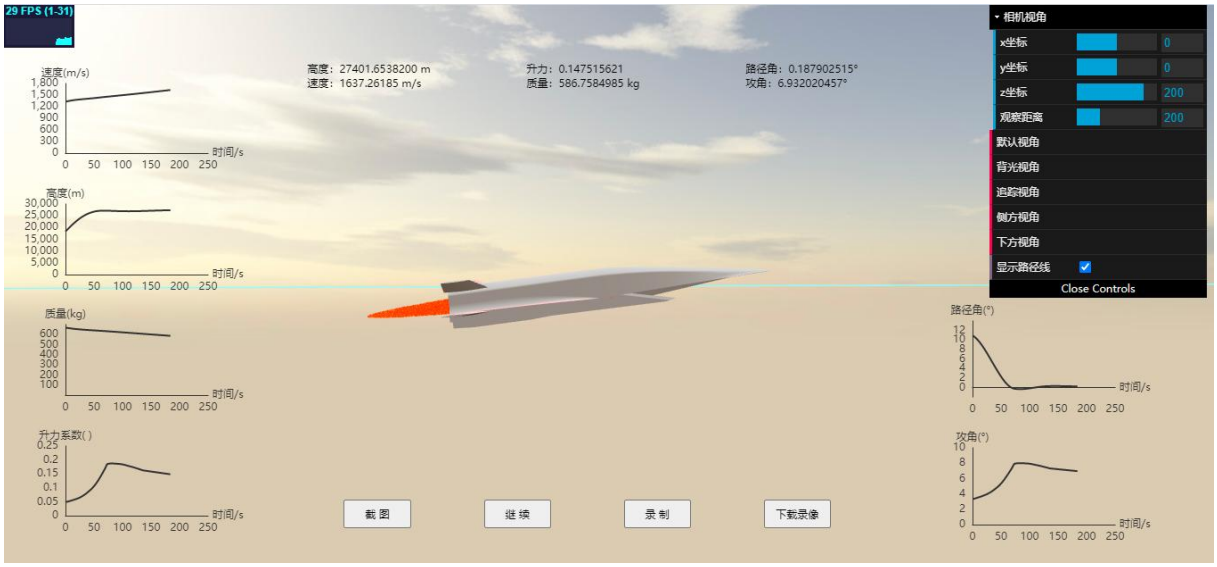

图 4-15 增添飞行器视角控制栏效果展示图

4.4.2 截图功能实现

截屏是电脑截取的视觉影像，可以在萤幕或其它显示装置上显示。为满足用户保存和共享飞行器某时刻飞行姿态的需求，该仿真系统增添了截图功能。用户在前端页面使用鼠标单击“截图”按钮，系统会自动截图用户当前观察到的图片，并在新的页面得到截取的图片。

当用户点击“截图”按钮，仿真系统执行自定义 shot()函数，并在该函数体内创建 Image 对象，然后通过调用 renderer.domElement.toDataURL()方法在浏览器端缓存当前所截取图片，并调用 window.open()方法在新的窗口展示当前浏览器缓存的图片。

代码 4-12 截图功能实现代码

```
function shot() {
                var image = new Image();
                var imgData = renderer.domElement.toDataURL("image/jpeg"); //这里可以选择
png格式jpeg格式
                image.src = imgData
                image.id = 'image'
                screen(imgData);
                if(!paused) pause()
            }
```

4.4.3 暂停和继续功能实现

为满足用户观察某一时刻飞行器的飞行姿态和气动数据需求，该仿真系统增添了“暂停”和“继续”功能。当用户使用鼠标单击“暂停”按钮时，Web 端停止对飞行器的渲染，飞行器仿真动画和气动数据可视化图表处于静止状态；当鼠标单击“继续”按钮时，系统继续对飞行器进行渲染，气动数据可视化表格实时更新。

首先定义 pause 标记变量并初始化为数值为 1，然后通过点击“暂停”和“继续”按钮实现 pause 数值转变。当 pause 为 1 时，飞行器坐标，尾焰粒子系统，气动数据参数以及天空盒渲染背景停止更新，以实现“暂停”功能；当 pause 为 0 时，与以上相反，以实现“继续”功能。

代码 4-13 暂停继续功能实现代码

```
function pause() {
                if(!paused) {
                    document.getElementById("play").innerHTML = " 继 续 "
                    paused = 1
                } else {
                    document.getElementById("play").innerHTML = " 暂 停 "
                    paused = 0
                    requestAnimationFrame(render)
                }
            }
```

4.4.4 录制视频实现

为方便存储和交流飞行仿真动画，系统向用户提供录制视频功能，当用户使用鼠标单击“录制”按钮时，系统对当前仿真页面进行录制，当再次单击“录制”按钮时，结束视频录制。当用户使用鼠标单击“下载录像”按钮时，系统会将之前录制的视频下载到本地计算机，方便飞行器设计相关人员存储飞行仿真视频，有利于之后的交流与探讨。

为满足用户使用录制仿真视频的应用需求，HTML5 提供的一个 MediaRecorder 录制接口，通过 MediaRecorder()构造方法进行实例化。一个新的 MediaRecorder 对象，对指定的 MediaStream 对象进行录制,支持的配置项包括设置容器的 MIME(Multipurpose Internet Mail Extensions)类型和音频及视频的码率。飞行仿真系统展示的是 canvas 录制，通过 canvas 的一个 API captureStream 获取录制对象的 MediaStream，从而实现录制视频。

代码 4-14 录制视频代码

```
function screen(base64URL) {
```

```
                var win = window.open(base64URL, '_blank')
                win.document.write('<a download="screenshot.jpg" onclick="download()"><img
id="shot" style="margin: -8" src="' + base64URL + '"></img></a>')
            }
```

4.4.5 下载视频功能实现

为方便用户缓存飞行仿真视频以方便互相交流学习的需求，仿真系统增添下载视频功能。通过系统后端直接把视频拼接起来，再通过一个请求推给系统前端这种方案在视频尺寸过大时，会导致 Callback 超时风险。因此可以通过 WebSocket 向前端不断地推小的视频切片，再由前端把将视频切片拼接起来，并借助浏览器转码实现视频下载。

代码 4-15 下载视频代码

```
function stopAndblobDownload() {
                recorder.stop();
                const link = document.createElement('a');
                link.style.display = 'none';
                const fullBlob = new Blob(allChunks);
                const downloadUrl = window.URL.createObjectURL(fullBlob);
                link.href = downloadUrl;
                link.download = `test${Math.random()}.webm`;
                document.body.appendChild(link);
                link.click();
                link.remove();
            }
```

4.4.6 检测仿真动画帧数实现

测试 3D 仿真动画性能好坏，可以使用帧数来衡量，帧数为图形处理器每秒钟能够刷新几次，通常用 fps(Frames Per Second)来表示，在 Three.js 中 ，性能由一个性能监视器来管理，首先，new 一个 stats 对象，然后将 stats 对象添加到仿真页面中，最后调用 stats.update()函数来统计时间和帧数。

代码 4-16 检测帧数代码

```
function initStats() {
                stats = new Stats()
                stats.domElement.style.position = 'absolute';
                stats.domElement.style.left = '0px'
                stats.domElement.style.top = '0px'
                document.body.appendChild(stats.domElement)
```

```
}
```

其效果实现图为图 4-15 中左上角部分。

4.5 本章小结

本章主要介绍了系统实现的细节部分，包括天空盒场景的构建，飞行器数据定义，模型加载，飞行姿态模拟，轨迹可视化，火焰粒子可视化，气动数据实时可视化以及截图，录像，下载等用户交互功能。

第 5 章 系统测试

5.1 功能测试

5.1.1 交互功能测试

表 5-1 为交互功能测试。

表 5-1 交互功能测试

测试编号	测试内容	测试数据	测试结果	通过状态
1	截图	点击截图	浏览器端页面展示所截取图片	通过
2	暂停	点击暂停	浏览器端停止对飞行器渲染	通过
3	继续	点击继续	浏览器恢复渲染	通过
4	录制	点击录制	系统开始对仿真动画录制	通过
5	录制	再次点击录制	系统终止对反正动画录制	通过
6	下载录像	点击下载录像	将之前下载视频下载到本地	通过
7	X 坐标	单击 X 坐标不放，移动鼠标	飞行器实现沿 X 轴方向移动	通过
8	Y 坐标	单击 Y 坐标不放，移动鼠标	飞行器实现沿 Y 轴方向移动	通过
9	Z 坐标	单击 Z 坐标不放，移动鼠标	飞行器实现沿 Z 轴方向移动	通过
10	观察距离	单击观察距离不放，移动鼠标	飞行器实现观察距离变化	通过
11	默认视角	单击默认视角	观察视角调整到默认视角	通过
12	背光视角	单击背光视角	观察视角调整到背光视角	通过
13	追踪视角	单击追踪视角	观察视角调整到追踪视角	通过
14	侧方视角	单击侧方视角	观察视角调整到侧方视角	通过
15	下方视角	单击下方视角	观察视角调整到下方视角	通过
16	显示路径线	单击显示路径线	浏览器端不显示飞行器轨迹	通过
17	Close Control	单击 Close Control	视角控制栏关闭	通过

5.1.2 可视化功能测试

表 5-2 为可视化功能展示结果。

表 5-2 可视化功能展示结果

展示编号	展示内容	展示结果
1	速度-时间曲线	正常展示
2	高度-时间曲线	正常展示
3	质量-时间曲线	正常展示
4	升力系数-时间曲线	正常展示
5	路径角-时间曲线	正常展示
6	攻角-时间曲线	正常展示
7	火焰粒子	正常展示
8	飞行器姿态模拟	正常展示
9	飞行器轨迹	正常展示
10	视角控制栏	正常展示
11	各气动数据数值	正常展示
12	帧数监视	正常展示
13	天空效果	正常展示

5.2 性能测试

以 X-51A 飞行器的飞行过程为例进行仿真性能统计。仿真的硬件平台为普通的个人计算机，处理器是 Intel(R) Core(TM) i5-4300U CPU @1.90GHz，8G 内存；测试浏览器版本为 GoogleChrome 91。借助 stat.js 监控该仿真结果的 FPS 信息如图 5-1 所示。

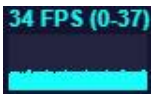

图 5-1 飞行仿真系统的 FPS 检测结果

经模拟过程完全统计，结果如图 5-2 所示。该飞行器仿真系统 Web 端页面每秒更新帧数均大于 30 帧，高于人眼分辨率的 24 帧。

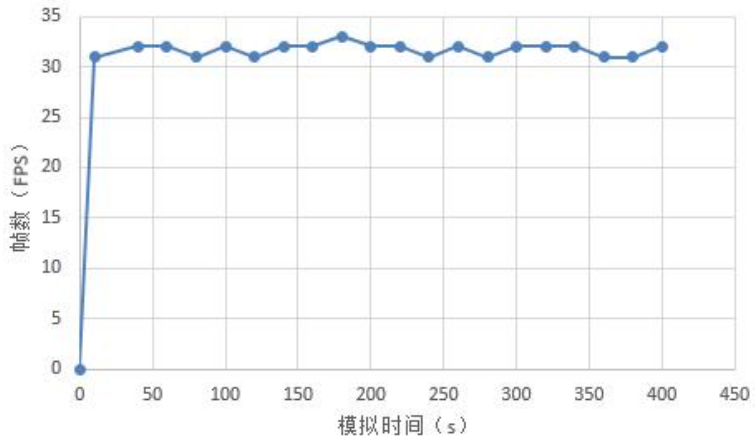

图 5-2 模拟飞行过程的帧数统计

5.3 功能实例

以 X-51A 相似飞行器飞行的实际数据，验证本文开发的系统软件功能。从视觉上可正常观察，飞行器虚拟飞行动态过程。界面中主体内容分布为：上部分为实时飞行状态数据值、左右两侧为状态时间变化的曲线、中间是飞行器虚拟飞行的运动过程；其次左上角与右上角分别是性能窗和视角控制窗，均可隐藏。

具体地，如图 5-3 是上升段画面，对应图 4-6 的阶段①，可以看见飞行器喷出火焰，向上爬升，从页面中高度与时间图像可以看出，飞行器高度在持续增加，因此可以判断飞行器处于爬升阶段。

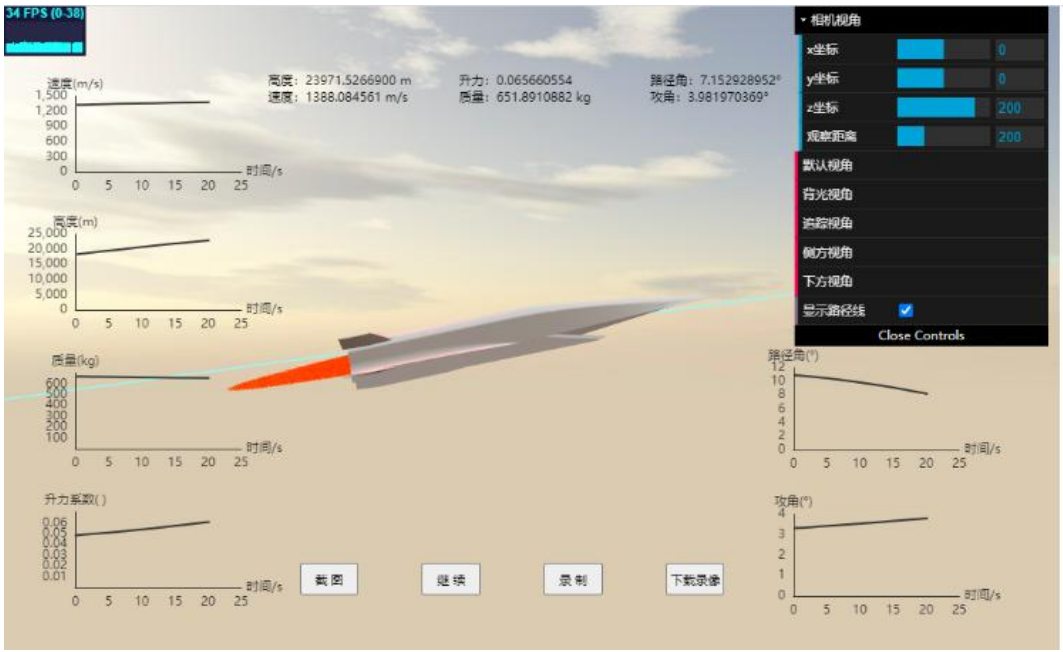

图 5-3 飞行器爬升阶段

图 5-4 飞行器处于上升段到下降段的过度过程，对应图 4-6 的阶段②，从页面中可以高度与时间图像可以看出，飞行器高度在一段时间处于稳定状态，因此可以判断飞行器处于巡航段。

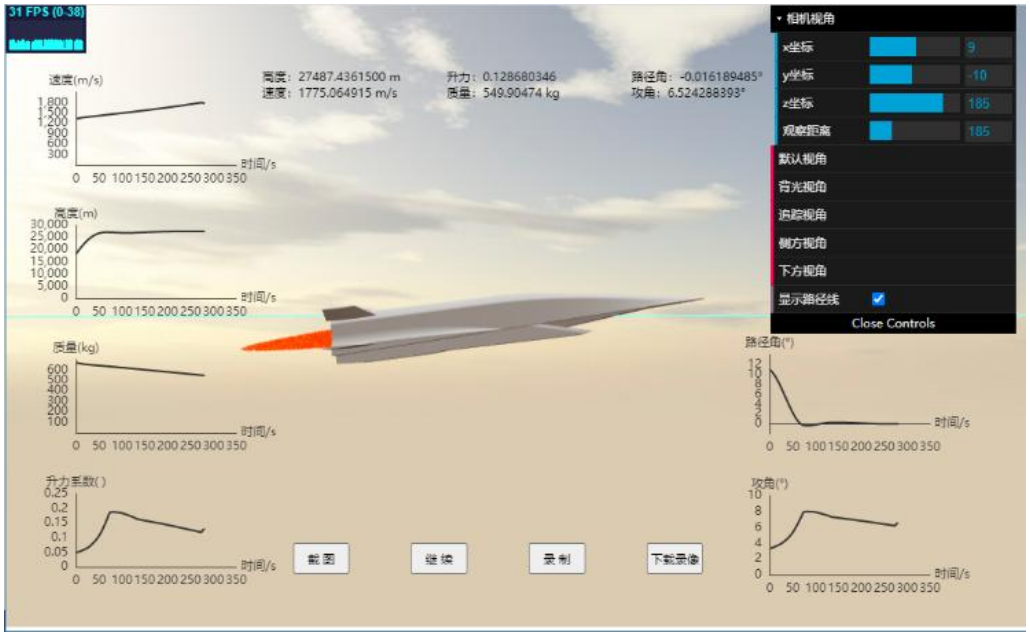

图 5-4 飞行器巡航阶段

图 5-5 是飞行器无动力滑翔下降阶段，对应图 4-6 的阶段③，从图中可以看出飞行器的姿态处于下降段，也可以通过高度与时间图像，可以看出飞行器高度随时间的增加，高度在不断减少，因此可以判断飞行器处于下降段。

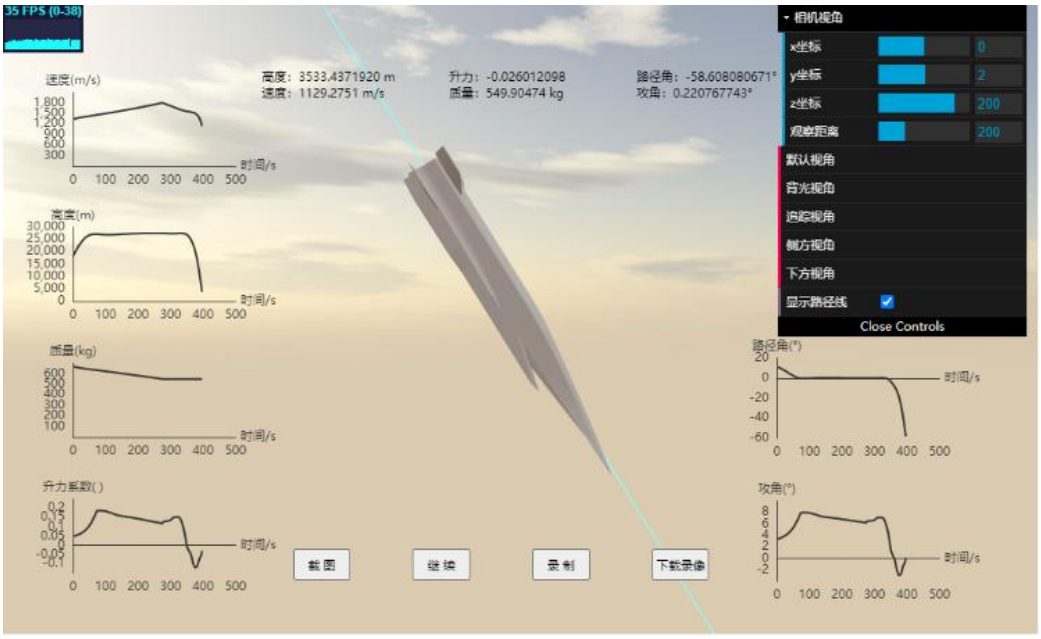

图 5-5 飞行器无动力滑翔下降段

图 5-6 是从默认视角，背光视角，追踪视角，侧方视角，下方视角观察飞行器的效果截图。

图 5-6 飞行器视角控制

可见，从功能上可见本文所设计的系统，能够准确的仿真 X-51A 相似飞行器的虚拟飞行过程，为该飞行器的可飞行性、飞行可达性提供了直观的过程分析工具。

5.4 本章小结

本章主要介绍了飞行器仿真系统功能测试及测试结果，包括交互功能测试和可视化功能展示以及各功能实例展示。

结论

课题针对一体化飞行器的飞行特点与虚拟仿真飞行需求，根据当前主流计算机设备能力，设计了一体化飞行器在线模拟飞行仿真系统。该系统能够与团队前期研究的一体化飞行器轨迹计算算法无缝衔接，以形成云端计算、客户端可视交互的在线虚拟仿真软件。

课题针对传统的飞行仿真依赖于硬件，对硬件要求高，专业性强的问题，提出了一种基于 Web 端的飞行仿真系统，该系统以气动数据为驱动，实时模拟了飞行器飞行的姿态和各个视角，实时可视化了气动数据，同时兼顾天空效果和尾焰特效。系统为用户提供了交互功能，如截图，下载录像等，方便了研究人员之间的协同交流合作。

本系统能够为计算机软件辅助飞行器设计方面提供技术参考。其意义具体表现为研制成本上的节约，分析能力方面的直观高效，研究人员研发协作能力的提升。

致谢

时光荏苒，白驹过隙。仿佛昨天还带着大一时上大学的喜悦，今天就要步入社会。回忆大学四年学习生活的点点滴滴，感慨万千。在大学四年的美好时光中，不仅仅学到了专业知识，更学会了独立思考。大学四年，老师的细心教导，同学朋友的相互帮助，家人的无私支持，都使我受益良多。

首先，我要感谢我的指导老师陈立伟导师，陈老师思想开放，思维缜密，学识渊博、沉稳大气、待人接物不卑不亢，无论是在他身上的研究精神还是生活态度，都是我们值得学习的地方。在大四这一年中，从论文开题，研究开展，中期答辩，最终答辩以及论文的撰写，陈老师都全程参与，亲自指导，给与了我极大的指导，帮助我顺利完成本科毕业设计。

其次，我还要感谢黄俊老师，黄老师不辞辛劳的指导，在系统构想、设计和实施过程的不吝指教，让我少走了许多弯路，对我能够在有限的时间内完成课题任务起到了决定作用，也是黄老师的帮助，让我成长了很多。

最后，我要感谢我的家人，是他们长期以来的无私奉献与支持和深切期望，给予我前进道路上的勇气和动力。

参考文献

[1] 钱锋. 加快工业软件研发夯实制造业高质量发展基础 [J]. 中国科技产业, 2021, (3): 30-1.

[2] TYREE M T, ZIMMERMANN M H. Xylem Structure and the Ascent of Sap [M]. Xylem Structure and The Ascent of Sap, 2002.

[3] MARZOLINI C, PAUS E, BUCLIN T, et al. Polymorphisms in human MDR1 (P-glycoprotein): recent advances and clinical relevance [J]. Clinical Pharmacology & Therapeutics, 2004, 75.

[4] 孙家广. 高端工业软件打破国外垄断抢占竞争制高点 [J]. 科学中国人, 2019, (7): 38-9.

[5] 王江峰, 王旭东, 李佳伟, et al. 高超声速巡航飞行器乘波布局气动设计综述 [J]. 空气动力学学报, 2018, 36(5): 705-28.

[6] 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): 8-12,7.

[7] 付沂辰. 导弹飞行视景仿真系统设计 [D]. 哈尔滨工程大学, 2016.

[8] 何晓骁, 王秉涵. 美军"实况-虚拟-构造"仿真技术发展及应用研究 [J]. 航空兵器, 2021, 28(6): 5.

[9] 康海龙, 范亚楠, 李斌. NASA 开源软件实践与思考 [J]. 卫星与网络, 2018, (4): 59-61.

[10] 王远明, 卢宽, 贾倩, et al. 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): 2087-92.

[11] 沈洋. 基于 OpenGL 的虚拟航空仪表的研究与实现 [D]. 华中科技大学, 2014.

[12] 叶舸, 田兆锋, 闫楚良. 基于 OpenGL 的飞机飞行实测数据可视化研究 [J]. 航空学报, 2011, 32(6): 1050-7.

[13] 闫晓东. 基于 OSG 的飞行视景仿真平台开发 [J]. 计算机仿真, 2008, 25(5): 58-60,198.

[14] 冯姣, 刘志勤, 黄俊, et al. 基于 Three.js 的飞行仿真系统设计 [J]. 计算机测量与控制, 2020, 28(2): 216-9.

[15] 胡亚海, 彭晓源, 王行仁. 基于 WEB 的仿真中的想定管理 [J]. 系统仿真学报, 2002, 14(3): 403-5,8.

[16] 沈宏伟. 基于 Flask 的企业内网安全系统的设计与实现 [D]. 北京交通大学, 2019.

[17] 罗彪, 张宏涛. 基于 HTML5 的移动互联网应用发展趋势 [J]. 信息与电脑（理论版）, 2018, (10).

[18] 朱晓飞, 万哲. 基于 OpenGL 的三维场景的模拟 [J]. 电子技术与软件工程, 2014, (3): 103.

[19] 周正, 贺旭照, 吴颖川. X-51A 飞行器模型的建立及性能初步研究 [Z]. 第十五届全国激波与激波管学术交流会论文集. 杭州. 2012: 266-70

[20] The X- 51A Scramjet Engine Flight Demonstration Program [Z]. 15th AIAA International Space

Planes and Hypersonic Systems and Technologies Conference. Dayton, OH(US). 2008.

[21] 黄俊, 刘知贵, 刘志勤, et al. 一体化飞行器面向控制的建模与弹道规划 [J]. 弹箭与制导学报, 2020, 40(1): 77-82.

[22] REEVES W T. Particle systems—a technique for modeling a class of fuzzy objects; proceedings of the Conference on Computer Graphics & Interactive Techniques, F, 1983 [C].

攻读本科学位期间取得的研究成果

攻读本科期间所获的软件著作权：

软件名称：一体化飞行器在线模拟飞行仿真系统 V1.0，登记号：2022SR0586809

攻读本科期间参研的项目：

1.计算机学院基地实践项目“在线数据拟合工具开发”，项目编号：20192007

2.西南科技大学创新基金“一体化飞行器在线模拟仿真飞行”，项目编号：JZ20-017

3.四川省创新基金“一体化飞行器在线模拟仿真系统”，项目编号：202110619001