Software Engineering and Applications 软件工程与应用, 2018, 7(3), 188-199
Published Online June 2018 in Hans. http://www.hanspub.org/journal/sea
https://doi.org/10.12677/sea.2018.73022

Hans汉斯

Realization of Collaborative Design Scene Based on Web

Jingjun Wang[1], Xiuli Shao[2], Huichao Li[2], Mengmeng Yao[2]

[1]Tianjin Baili Ertong Machinery Co., Ltd., Tianjin
[2]Department of Computer and Control Engineering, Nankai University, Tianjin
Email: zhb@tet.cn, shaoxl@nankai.edu.cn

Received: Jun. 9th, 2018; accepted: Jun. 22nd, 2018; published: Jun. 29th, 2018

Abstract

Product design work often requires many people to participate in different places. Therefore, this article designs and implements a Web-based product collaborative design system, which realizes the construction and restoration of a design scenario for a collaborative product, the loading of parts libraries, and the addition and editing functions of part models in the scene, dialogs in the design process, various documents, and interactive design features for the product.

Keywords

Collaborative Design, Scene Creation, Scene Recovery, WebGL

基于Web的产品协同设计场景的实现

王景军[1]，邵秀丽[2]，李慧超[2]，姚萌萌[2]

[1]天津百利二通机械有限公司，天津
[2]南开大学，计算机与控制工程学院，天津
Email: zhb@tet.cn, shaoxl@nankai.edu.cn

收稿日期：2018年6月9日；录用日期：2018年6月22日；发布日期：2018年6月29日

摘　要

产品设计工作往往需要多人异地参与，因此，本文设计实现了基于Web的产品协同设计系统，实现了针对某一协同产品的设计场景的构建与恢复、零件库的加载、场景中零件模型的添加与编辑功能，设计过程中的对话、各种文件，以及在线对产品的交互设计功能。

文章引用: 王景军, 邵秀丽, 李慧超, 姚萌萌. 基于 Web 的产品协同设计场景的实现[J]. 软件工程与应用, 2018, 7(3): 188-199. DOI: 10.12677/sea.2018.73022

关键词

协同设计，场景创建，场景恢复，WebGL

1. 引言

智能制造需要协同设计的支持，这在目前已成为一个重要的研究热点和应用热点。协同设计技术在协同 CAD，工业设计等制造设计中有着广泛的应用[1] [2]。借助产品协同设计平台，异地人员可以参与产品的设计工作。其中，基于 Web 3D 的协同平台是其中重要的组成部分[3] [4]。

基于 Web 3D 的计算机协同研究主要的研究点和难点有：1) 3D 模型和场景在协同环境中的表示形式和传输方式；2) 实现不同用户多视角、多精度的灵活配置方式；3) 3D 场景的保存与恢复。

为了能够实时展示 3D 图形并进行可视操纵，需要有合适的 3D 场景与模型展示与操纵界面，本文系统采用 B/S 结构，采用封装了 Web GL [5] [6]的 Web 3D 交互技术 Three.js [7]，设计 3D 场景操作集合，实现了 3D 设计场景的整体保存与恢复以及零件拼装设计的功能。设计人员可将线下设计的 STL 格式的 CAD 模型直接导入 3D 场景中进行拼装；用户可以调整 3D 模型大小、位置、旋转角度，可以清空场景、删除选中物体，进行上下步操作回滚等。

2. 基于 Web 3D 的设计场景设计

参与设计者在设计界面进行零件选择或者上传线下做好的 CAD 零件模型，在 3D 场景中完成零件的基本编辑操作，同时支持场景中零件的移除、场景的清空、上下步操作的回滚[8]、设计过程中历史版本的保存和恢复以及“后来者”可同步之前设计结果等功能，以更好的支持用户在 3D 场景中的产品设计活动。最终，完成零件模型在场景中拼装成整车的个性化设计过程。

整个设计界面包含了在 3D 场景下进行产品拼装设计的所有操作集合，用户单步操作完成，操作数据交由服务器去进行操作的冲突判定或者同步控制，整个协同设计场景的界面设计图如图 1 所示。

2.1. 拼装产品零件模型的来源

1) 平台后台进行零件库的维护，该零件模型应当是.obj 模型文件结合.mtl 材质文件格式；

2) 参与设计用户自行导入 CAD 模型文件，该文件一经导入可直接加载至 3D 场景中，即提供外来建模零件接收接口，格式限定为 STL 格式。

2.2. 零件模型在 3D 场景中的编辑操作

1) 零件颜色的更换，用户在 3D 场景中选中要进行颜色更换的零件模型，并选择要更换的颜色，进行零件模型颜色的更换；

2) 3D 场景中零件的基本编辑，用户在 3D 场景中选中进行编辑的物体，并设置编辑模式，模式的设置提供 UI 形式切换和键盘快捷键式切换两种方式。编辑模式有：大小(scale)、位置(position)、旋转角度(rotation)，分别同键盘的 R，W，E 对应。在对应模型下，通过鼠标在对应方位上的拖拽完成对应编辑操作，或者直接通过设计界面零件对应坐标、大小、旋转角度参数去编辑模型零件。

Figure 1. Collaborative design scenario interface design draft
图 1. 协同设计场景界面设计草稿

2.3. 3D 场景中设计过程的操作

1) 场景基本操作：a) 场景的清空操作、b) 场景中某个零件模型的删除、c) 操作上一步动作的回滚、d) 操作下一步动作的回滚等；

2) 历史记录的保存与恢复：对 3D 场景中的物体进行设计操作过程中，可随时对当前设计结果进行保存，并以“对象名称+当前时间格式”命名规则在设计界面历史记录中显示，用户点击对应记录可恢复至历史设计状态。

3. 3D 场景设计操作的实现

为实现用户 Web 3D 场景设计解决方案，为用户提供涵盖网页端的三维模型展示、设计场景交互、三维零件模型编辑等功能的在线网页端产品设计，下面分别从 3D 交互场景的构建、零件库的加载、零件模型的添加与编辑、设计过程支持基本操作等方面介绍 3D 场景设计的实现工作。

3.1. 3D 交互场景构建

3D 交互场景为参与设计的用户提供一个在浏览器中三维实时展示以及场景中进行交互设计的环境。图 2 给出了本文 3D 交互场景的构建工作流程。首先，在网页端创建场景，并构建了支持场景中三维模型展示功能的基本组件，包括：相机、底部参照网格、光源、显示渲染器等组件的添加；其次，在基本组件的基础上，构建了支持场景中模型的鼠标选中交互、场景中物体的 360 度查看和模型大小、位置、旋转角度的编辑功能；最后，开启线程，实时监测场景状态变化并实时进行场景中组件的渲染显示。

实现中运用 javascript 网页端编程借助 Three.js 框架提供的类库展开对 Web 3D 交互场景的构建工作，其中，Three.js 框架完成了 WebGL (Web Graphics Library) [9] [10]的封装，其中场景中组件的添加借助场景对象的 add()方法完成。

交互场景构建工作的具体实现流程如下：

第一步：场景与场景的内部构件创建

1) 创建初始化场景，使用 Three.js 类库的 THREE.Scene()方法创建初始化场景对象 scene；

Figure 2. Scene construction flow chart
图 2. 场景构建流程图

2) 相机的添加，使用类库的 THREE.PerspectiveCamera()方法创建场景中的远景相机，指定场景查看角度，设置相机摆放位置 position，并通过 lookAt()函数设置函数观察点，最后将其添加到 1)场景对象中，即完成场景中模型的添加；

3) 参照网络的添加，使用类库的 THREE.GridHelper()方法创建场景网格，通过传递参数设置网格的尺寸、两条线的间隔、中心线条以及其他非中心线条的颜色，并设置网格位置，本次设置场景的中心位置坐标(0,0,0)的位置，最后将其添加到场景中。

第二步：场景中的光源的设置

1) 环境光的添加，通类库的 THREE.AmbientLight(0x444444)方法创建环境光对象，其中参数为实例化好的 THREE.AmbientLight 对象颜色(占位符 1)，本次采用灰色环境光，创建完后将环境光添加到场景中；

2) 平行光的添加，通过类库的 THREE.DirectionLight(0xffeed)创建平行光对象，其中参数为平行光颜色，并设置平行光方位，调用平行光对象的 set(20,20,20).normalize()进行设置，最后将平行光对象添加到场景中。

第三步：场景的渲染显示

场景的渲染在整个交互场景的构建工作中有重要的作用，决定了场景中组件以及模型的显示。使用类库的 THREE.WebGLRenderer()创建渲染器对象，使用 render()方法设定参数场景对象 scene 和相机对象 camera，对场景中的构件进行渲染显示，其后设置渲染清除颜色、设置场景显示大小、场景中物体是否排序等特性，最后将渲染器中的所有元素 renderer.domElement 添加到页面位置进行显示，即为页面显示的场景。

第四步：设计场景用户交互事件的添加

1) 场景中物体的 360 度查看，查看角度随鼠标的拖拽而改变。通过 THREE.OrbitControls(camera)方法进行旋转控制设置，为场景相机对象添加旋转属性，相机的查看视角可随鼠标的拖动而改变。

2) 添加零件模型的编辑事件，通过 THREE.TransformControls()方法创建编辑对象，设置参数 camera、renderer.domElement 为场景中渲染的物体添加编辑事件，设有 translate 位置、rotate 旋转、scale 大小编辑模式，场景中的物体在该三种模式下，会根据鼠标在固定方位上的拖拽进行对应编辑，默认为 translate 位置模式。

3) 添加场景中物体点击捕获事件，首先为渲染器设置 mousemove 事件和 mousedown 事件。其中 mousemove 是事件设置鼠标捕获物体图标为手型(正常情况下为箭头)；mousedown 事件通过 transformControls 编辑对象的 attach()方法设置鼠标选中物体，选中的物体可进行场景中设计提供的所有编辑操作，通过 transformControls.object 获取选中物体。

其中，鼠标选中物体需要解决鼠标从二维坐标空间到三维极坐标空间的转变问题，引入类库的 THREE.Raycaster()方法定义投射对象，使用类库的 THREE.Vector3()设置鼠标投射向量对象，通过调用投射对象的 setFromCamera(vector, camera)方法根据相机以及投射鼠标向量设置投射参数，并通过 intersectObjects(objects, true)找寻投射角度与鼠标的交叉点物体，即为鼠标捕获场景中物体。

4) 添加键盘快捷键事件，为浏览器窗体对象添加键盘的“keydown”事件，并通过对应 case 选项，对场景中 transformControls 编辑对象通过 setMode 方法更改焦点物体的编辑模式，其中，“W”、“E”、“R”键分别对应 translate 位置、rotate 旋转、scale 大小模式。

最后，建立线程实时对场景进行渲染

完成上述交互场景的创建后，调用 animate()方法对场景中的编辑操作实时显示。该方法的主要工作有：调用 requestAnimationFrame 方法，设置线程循环执行场景的渲染工作，包括对场景中相机的旋转角度的空间的更新、编辑物体的更新，通过调用 update()函数实现，并使用 renderer 对象的 render(scene, camera)方法进行场景中所有组件的渲染。

3.2. 产品零件库的加载

参与设计人员进入设计界面，设计界面自动进行零件库的加载显示，供用户选择。在进行协作设计前，需平台后台或生产厂商进行零件库的维护。

1) 零件的存储

零件的存储分两部分：a) 零件模型、零件简图(为在页面显示做准备)存入服务器文件区，其中零件模型包括.obj 文件以及其对应的默认.mtl 材质文件；b) 零件记录，如零件名称、模型名称、简图名称、零件类型等信息存入数据库中，数据库中的表数据记录跟文件区存储的模型和零件简图一一对应。

2) 零件库的加载

参与设计用户进入设计界面，页面后台写好 sql 语句，调用数据库访问接口，读取数据库零件库信息，获取零件模型信息、零件简图 url，并将零件简图采用后台的 InnerHtml 方法按零件类别分别填充至设计页面零件区的不同位置，同时为零件简图添加选中载入设计场景事件，供用户进行零件模型的选择。

3.3. 3D 零件模型的添加与编辑

在 3D 场景中进行的产品模型的拼装设计过程，为方便记录当前设计结果，参与客户端定义本地设计对象副本结构变量，来记录当前编辑产品模型的设计状态[6]。同时协同成员之间通过维护设计对象副本的一致性，来维护场景中设计结果的一致性[7] [11]。因此，用户单步操作操作完成，需要先进行本地

设计对象副本的更新，然后进行协作的操作的冲突判定或同步控制。

本节零件模型的操作从场景中零件库模型的添加、STL 零件模型的导入、零件模型颜色的更换三部分展开介绍。

3.3.1. 零件库模型的添加

图 3 给出了整个零件库模型的添加过程的具体过程。即用户点击零件区的某一零件模型，到服务器固定文件夹下读取对应.obj 模型文件以及默认.mtl 格式材质文件，并将合并材质后的模型文件载入到场景的随机位置中。其后，更新本地设计对象零件副本，新增信息发送到 NodeJs 服务器进行协作组内操作的冲突判断或是同步处理。

首先，材质的加载。通过 Three.js 类库的 THREE.MTLLoader()方法定义材质加载对象，使用 setPath('model/')设置材质文件读取路径，调用材质对象方法 mtlLoader.load(name,func)方法加载材质；

其次，场景中模型的载入。通过 Three.js 类库的 THREE.OBJLoader()方法创建.obj 模型文件加载对象，使用 setMaterials 方法为加载对象设定上一步加载材质，同样通过对象调用 setPath('model/')设置.obj 文件的读取路径，通过调用 objLoader.load(name,func,onProgress,onError)方法加载 obj 文件，并将加载的模型设置其在场景的位置(本次设定为场景 x，y 平面上的随机位置)，通过 scene.add(obj)载入场景中；

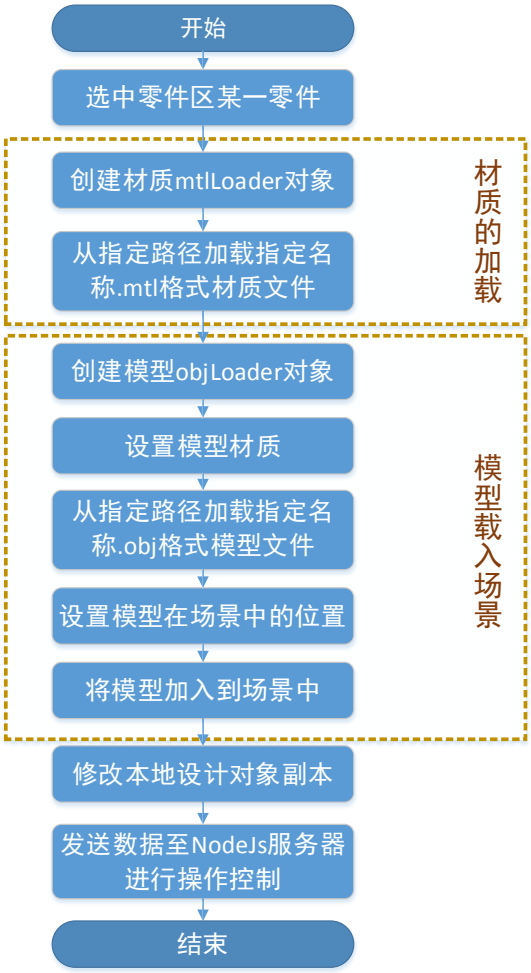

Figure 3. The process of adding a part model
图 3. 零件库模型的添加流程

最后，修改设计对象副本 data，定义新增零件结构 newObjData 并将其添加到设计对象副本 data 中，该结构包含零件名称、零件标志(零件库还是自行上传的 STL 文件)、操作用户名、零件位置、旋转角度、大小等零件属性。更改后，将新增零件信息使用 socket.io 即时通信类库中的 socket.emit 方法将零件属性发送至 NodeJs 服务器，供服务器进行操作冲突的判断或同步的处理。

3.3.2. STL 零件模型的导入

图 4 给出了 STL 零件模型导入的具体实现过程，即设计参与者在协同进行产品设计过程中，可随时将外部零件模型导入场景中，为产品整车设计提供个性化零部件素材，并实现单用户导入协作组内客户端用户同步可现。其中，零件模型的格式限定为 STL 格式，该格式为 CAD 软件、3Dmaxs 软件、Three.js Web3D 接收的通用版本。

用户选择上传零件模型文件，系统先对零件格式进行验证，验证通过进行零件模型的导入工作。

第一步：读取上传文件，并完成到 STL 模型对象的转换。创建 FileReader 对象，将上传文件 file 传

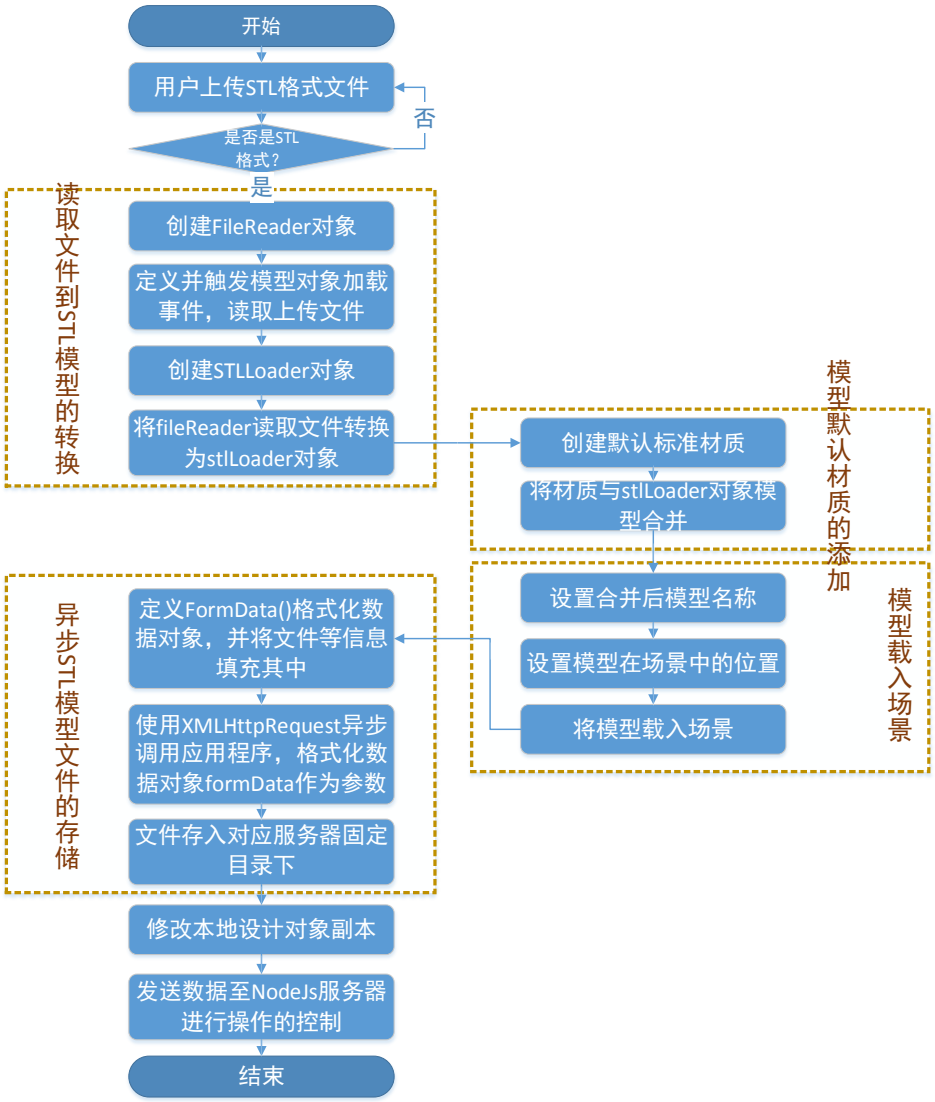

Figure 4. The process of importing STL model
图 4. STL 模型导入流程

入对象的 readAsArrayBuffer(file)方法中，触发对象的加载事件，将载入的文件通过 Three.js Web3D 类库的 STLLoader 类的 parse(conetents)方法将文件转换为 STL 模型对象；

第二步：为模型对象添加默认材质。通过 Three.js [7]类库提供方法 MeshStandardMaterial()创建标准材质实例对象，然后调用类库中的 THREE.Mesh()方法，以模型文件对象和材质对象作为参数，返回合并材质后的 STL 模型对象；

第三步：将合并材质后的 STL 模型对象载入场景。在载入场景前，先设置模型对象的名称 name，在场景中显示的位置 position (本次设定为场景中 x，y 平面上的随机位置)，并通过场景提供的 scene.add(mesh)方法，以合并后的 STL 模型对象作为参数，将模型对象载入场景中；

第四步：将上传的模型文件存入服务器文件区。创建 FormData 格式化数据对象，将上传文件以及文件名称使用 append()方法加入到 formData 对象中。创建 XMLHttpRequest 异步请求对象，并指定异步请求方式(本次为 post 方式)、异步请求地址。以 formData 格式化数据对象作为参数，传递到指定地址应用程序，应用程序定义 HttpPostedFile 对象来接收文件参数 file 及字符串对象接收文件名称 filename，并使用该对象的 SaveAs('../model/upload/'+filename)方法，传入文件保存地址，将文件保存到服务器的指定位置；

最后，更新设计对象副本 data，修改后向服务器发送新增零件信息，服务器进行操作冲突的判断或同步的处理(本部分同零件库模型的加载)。

3.3.3. 零件模型颜色的更换

首先，获取页面颜色插件选中颜色 rgb 值，采用 transformControls.object 获取场景中当前操作物体；其次，采用 Three.js 类库中的 THREE.MeshPhongMaterial()方法通过传递 color 颜色参数，创建颜色材质对象；最后，将 object 物体的 material 属性，设置为该颜色材质对象。颜色更换结束，更新本地设计对象数据副本，并将模型颜色以及模型名称进行封装，发送至 NodeJs 服务器，供服务器进行操作冲突的判断或同步的处理。

3.3.4. 场景中的零件清除

为给参与设计提供一个灵活、可回溯的设计过程，本次在 3D 场景中完成的产品拼装设计操作，除对产品零件模型提供载入、大小、位置、旋转角度、颜色等的设计操作外，还提供了设计场景中零件的删除、场景的清空、场景中上下步操作的回滚等功能。

1) 删除选中零件

通过 transformControls 对象的 object 属性获取场景中当前操作物体，采用场景对象的 scene.remove(object)方法将选中物体从场景中移除，同时移除本地设计对象数据副本对应零件信息，将移除零件模型封装后发送至 NodeJs 服务器，供其进行冲突的判断或同步操作的处理。

2) 清空场景

获取本地设计对象数据副本，逐条遍历副本数据的零件信息，取出零件名称 data.modelSign，通过 scene.getObjectByName(data.modelSign)方法获取场景中名称对应物体，并使用 scene.remove(obj)方法，以获取物体作为参数，将该物体从场景中移除。最后，清空本地设计对象副本，向 NodeJs 服务器发送清空场景信息，供服务器进行同步用户操作的处理工作。

4. 3D 场景的恢复问题的解决

4.1. 历史记录的保存与恢复

在协作设计过程中，拥有设计界面操作控制权的用户可随时对当前设计结果进行保存，并随时将场

景中当前设计状态恢复至某一历史保存结果状态。

1) 历史记录的保存

前置条件：用户在场景中进行拼装设计过程中，需要对当前场景中的设计状态进行保存，点击“保存”保存当前设计成果。

应用程序解决方案：本次该过程充分使用本地设计对象副本数据，获取当前设计副本结果数据data.modelObjs 并将其转化为结果字符串 dataStr，将结果字符串采用 Ajax 异步请求方式在数据库中间结果表中进行永久存储(为中间结果的查询以及“后来者”同步设计状态做准备)，存储成功后在设计界面历史记录列表中添加以“结果名称+日期格式”命名的结果记录，并将结果字符串 dataStr 隐式存入记录中，然后为该条记录添加点击进行场景的恢复事件，为历史记录的恢复做准备。最后设置保存命令标识，将保存数据发送至 NodeJs 服务器端供其进行同步控制。

2) 历史记录的恢复

前置条件：用户在场景中进行拼装设计，需要对将场景恢复至某一历史记录结果状态，在此基础上进行再设计，在设计界面历史记录区点击要恢复的历史记录，进行场景中历史结果的恢复。

应用程序解决方案：获取记录中隐藏结果字符串 dataStr，先根据当前设计对象副本数据逐个将设计场景中的零件模型移除，即清空场景。然后，将结果字符串转换为 JSON 数据对象，逐条遍历对象中的零件数据，根据零件的属性值进行零件在场景中的恢复。最后设置恢复命令标识，将恢复结果数据发送至 NodeJs 服务器端供其进行同步控制。

4.2.“后来者”同步问题的解决

基于协作活动的特点，协作设计活动的一次创建可进行多次迭代设计，直到组织者结束该活动为止。由此，在协作成员进行非第一次协作设计的迭代时，需要同步获取之前活动的交互信息；或者一个用户加入一个已经存在的协作设计过程、再或者当系统的软、硬件、网络发生故障时，程序被迫退出后的重新进入，都会引发该同步问题。为此，给出解决方案。首先，在用户进行设计过程中，单步操作完成或进行文字、文件交流通信时，采用异步请求的方式实时将结果数据在数据库中进持久化存储；其次，用户进入协作设计页面，后台会自动从数据库获取设计活动当前结果数据，若结果不为空，则根据数据信息，进行文字、文件交互信息以及设计界面设计状态的恢复。具体的实现过程分交互结果的实时存储和及时恢复两部分进行介绍。

1) 交互结果的实时存储

设计界面需要存储的信息包括：3D 场景设计状态信息、文字文件交流信息、中间历史记录信息，其中，中间历史记录信息的存储在上一节历史记录的保存中已经介绍。3D 场景设计状态信息的实时存储，具体实现流程图入下图 5 所示。

页面驱动程序捕获用户设计操作，单步操作完成将本地设计对象副本 data.modelObjs 更新，将更新后的 data.modelObjs 转化为字符串格式，采用 Jquery 的$.ajax 异步 post 请求的形式进行结果字符串数据的永久存储。

具体实现过程如下：完成本地设计对象副本 data.modelObjs 更新后，调用 JSON.stringify(data.modelObjs)方法，以对象副本作为参数将其转换为字符串格式，记为结果字符串；然后使用$.ajax 异步 post 调用.ashx 应用程序形式，将结果字符串结合协作活动 id 作为参数传递到应用程序，后台应用程序书写 sql 语句并调用数据库存储接口将结果字符串填入协作活动 id 对应协作活动的表记录中，进行活动当前设计结果的持久化存储。存储结果通过 Response.Write(sign)方法[12]将信息写入 HTTP 响应输出流反馈至前台调用处，其后解析存储结果，失败对用户进行提示。

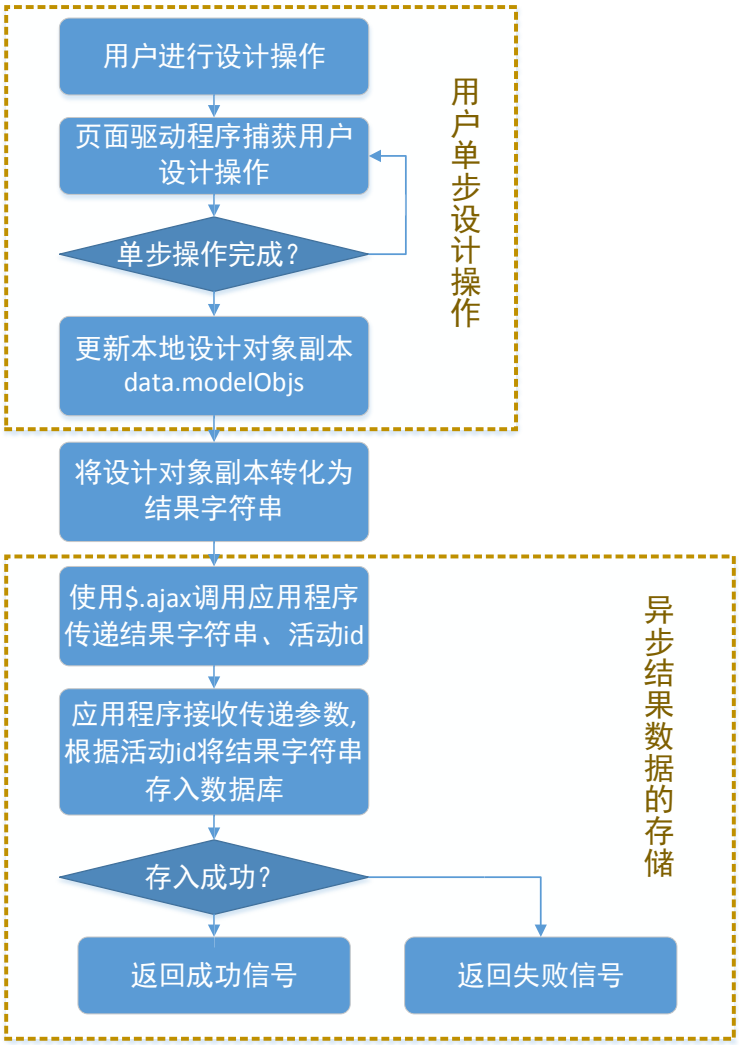

Figure 5. 3D scene design status real-time storage
图 5. 3D 场景设计状态实时存储流程

2) 交互信息的及时恢复

进行历史交互信息的恢复分别对 3D 场景设计状态的恢复、文字文件交互信息的恢复、历史中间结果的恢复三部分。

用户进入设计界面，后台应用程序分别按照协作活动 id，以及信息的存储状态作为过滤条件进行信息数据库的读取工作，读取的数据信息按照信息类别逐条进行交互状态的恢复。其中，3D 设计场景状态的恢复，以后台应用程序读取数据库的当前设计状态为结果字符串，进行场景设计状态的恢复，恢复过程同上一节历史记录的恢复过程。

4.3. 3D 场景中上下步操作的回滚

设计参与者在 3D 场景中进行设计操作过程中，可随时进行上下步操作的回滚。且当单用户完成操作的回滚，协作组内其他客户端同步进行回滚操作，内部工作示意图如下图 6 所示。

由图 6 可知，操作回滚的范围为产品设计的整个操作集合，在此基础上借助上一步和下一步栈空间进行上下步操作日志的存储。具体过程为：

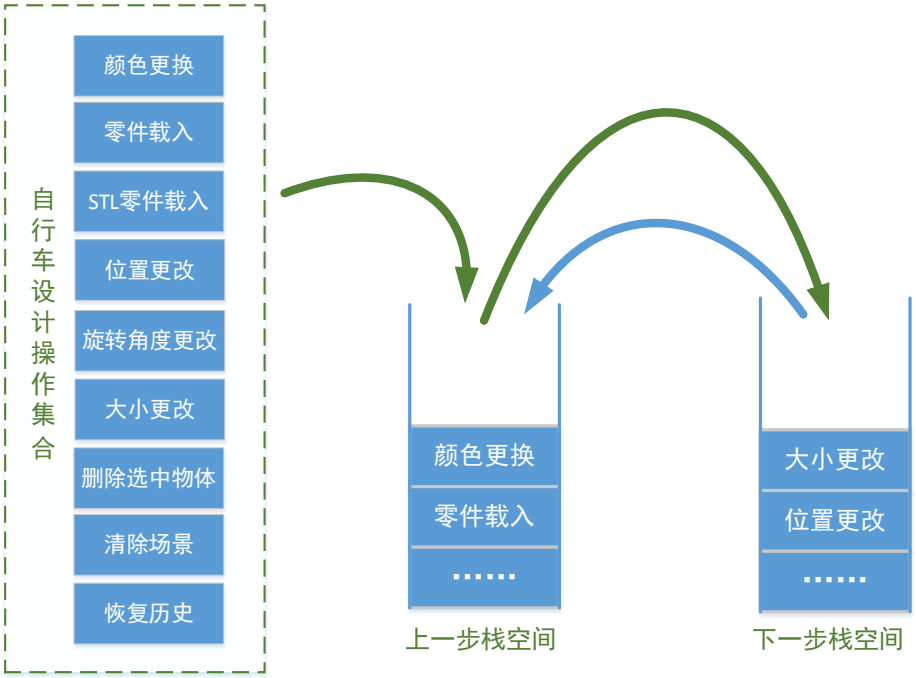

Figure 6. Step up and down to roll back the internal working diagram
图 6. 3D 场景设计状态实时存储流程

1) 用户单步操作完成将对应操作日志压入上一步栈空间，点击设计界面“恢复到上一步”按钮，进行上一步操作结果的回滚，具体是将上一步栈空间进行出栈，根据出栈日志去进行对应操作的回滚，同时将该回滚操作日志压入下一步栈空间，以供进行下一步操作的恢复；

2) 用户点击设计界面“恢复到下一步”按钮，进行上一步回滚操作的恢复，具体是将下一步空间进行出栈，根据出栈日志进行对应操作的恢复，同时将该操作日志再次压入上一步栈空间，以供用户进行设计操作的反复。

5. 结束语

本文主要在协同设计系统中引入 3D 场景的设计，设计实现了基于 Web 的产品协同设计系统，实现了针对某一协同产品的设计场景的构建与恢复、零件库的加载、场景中零件模型的添加与编辑功能。但仍有需要改进与研究的问题，例如：模型的标准化与语义管理，以及平台需要进一步应用验证，以及设计上应考虑一定的通用性等。

基金项目

天津市互联网先进制造专项(15ZXHLGX00360，15ZXHLGX00380)天津市重大科技专项(16ZXHLGX00250，15ZXDSGX00090)。

参考文献

[1] 殷国富, 陈永华. 计算机辅助设计技术及应用[M]. 北京: 科学出版社, 2000.

[2] 王新光. 基于模型驱动的实时交互式三维场景构建方法及其在水利上的应用研究[D]: [硕士学位论文]. 南京: 河海大学, 2002.

[3] 孔庆复. 计算机辅助设计与制造[M]. 哈尔滨: 哈尔滨工业出版社, 1999.

[4] 潘康华. 基于 MBD 的机械产品三维设计标准关键技术与应用研究[D]: [硕士学位论文]. 北京: 机械科学研究总院, 2012.

[5] Rego, N. and Koes, D. (2015) 3Dmol.js: Molecular Visualization with WebGL. *Bioinformatics*, **31**, 1322. https://doi.org/10.1093/bioinformatics/btu829

[6] Hanson, R.M., *et al.* (2013) JSmol and the Next-Generation Web-Based Representation of 3D Molecular Structure as Applied to Proteopedia. *Israel Journal of Chemistry*, **53**, 207-216. https://doi.org/10.1002/ijch.201300024

[7] http://www.wjceo.com/blog/threejs/2018-02-12/26.html

[8] Klein, M. (1989) Supporting Conflict Resolution in Cooperative Design Systems. *Artificial Intelligence in Engineering*, **21**, 1379-1390.

[9] Congote, J., Segura, A., Kabongo, L., *et al.* (2011) Interactive Visualization of Volumetric Data with WebGL in Real-Time. *Proceedings of the International Conference on Web 3D Technology*, Paris, 20-22 June 2011, 137-146. https://doi.org/10.1145/2010425.2010449

[10] https://www.khronos.org/webgl/

[11] http://www.hewebgl.com/article/articledir/1

[12] https://msdn.microsoft.com/zh-cn/library/system.web.httpresponse.write.aspx