

虚拟车间三维监控系统研究与实现

中国电子科技集团公司第三十八研究所 胡祥涛
东南大学 罗宏亮

虚拟车间三维监控系统是将虚拟现实技术应用到工业生产中的产物。本文提出了一种可行、高效的虚拟车间三维监控系统实现流程，并对其中的实现方法和关键技术展开详细的研究。在相关研究的基础上对虚拟车间三维监控系统进行设计，并详细讲解系统的实现方法。最后以车载雷达装配线为对象，验证本文提出的虚拟车间三维监控系统实现流程和方法的正确性、可行性。

一、简介

虚拟车间三维监控是通过建立生产现场的三维数字化模型，用生产数据驱动数字化模型，最终实现虚实同步的监控技术。随着建模与仿真技术、实时通信技术和 Web 技术的发展，虚拟监控系统逐渐兴起和发展，国内外学者对其进行了相关研究和实现。在国外，Senkuvienė 等提出了一种能实时监控车间状态和加工设备的可视化方案，针对生产进度、加工过程、资源调度等进行监控，并在 Stevila 公司部署应用，取得了良好的效果。Zoolfakar 等将仿真技术应用于生产制造过程，将研究成果开发成一个中间件，可以配合车间数据对生产进度进行仿真和监控。德国西门子公司开发了车间监控软件 Comos，可实现对生产现场的仿真、监控、场景漫游等，得到了市场的广泛认可。美国参数技术公司（PTC）推出了物联网平台 ThingWorx，该平台以网页技术为依托，具备强大的数据处理能力，开放了二次开发端口，可实现生产现场的三维可视化数字双胞胎。在国内，常仕军等应用甘特图等方式对生产现场的数据进行展现。李智等对数据的采

集方案进行研究，提高了数据的实时性，但展示界面还处于二维水平，缺乏真实性。柯榕等基于 Unity3D 平台构建了车间三维场景，但结构相对简单。李超等以塔式起重机为对象，建立了塔机三维模型和环境模型库，采用 DirectX 设计虚拟监控系统渲染引擎，实现了三维图形逼真显示和人机交互功能，但该方法工作量大，可迁移性低。梁兴明等通过 Unity3D 平台实现了虚实融合的车间实时生产监控，但着重细节研究，系统性不足。近五年来，国内企业对虚拟监控的需求越来越强烈，但由于国内在相关方面研究尚浅，还未形成完善的整体架构和成熟的技术理论，亟需一种系统、高效、适合市场应用的虚拟监控实现流程和方法。

二、虚拟车间三维监控实现流程研究

虚拟车间三维监控是通过三维虚拟方式展示车间的实时运行状态。运用三维建模软件对车间设备、物料等生产要素进行三维建模，采用虚拟现实引擎进行虚拟车间的渲染与交互动作建模，最后运用数据交互技术实现虚拟场景与生产现

场的实时同步,实现虚拟车间三维监控,本文提出的虚拟车间三维监控系统实现流程如图1所示。

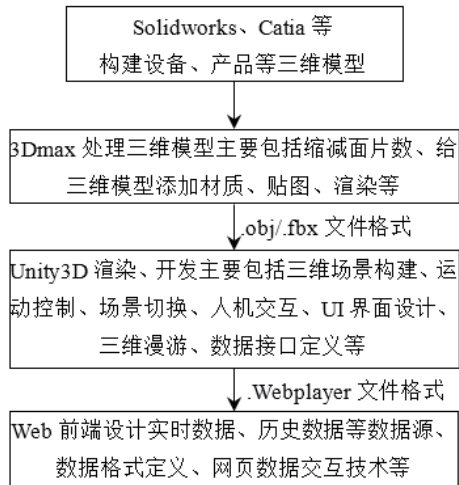


图1 虚拟监控实现流程

1、三维模型构建

三维模型是虚拟车间三维监控系统实现的基础。常见的三维建模软件由CATIA、Pro/ENGINEER、SolidWorks、Maya等,它们共同的特点是利用基本的几何元素,如球体、圆柱体、立方体等,通过拉伸、旋转、平移以及布尔运算等几何操作来构建复杂的几何模型。这些三维建模软件构建的三维模型主要侧重于对象的几何外形和各零部件之间的装配关系等信息,与现实世界中形态各异、丰富多彩的物体还存在一定差距。

2、三维模型处理

一般来说,采用三维建模软件构建的三维模型主要是对象的几何外形和各部件之间的装配关系,结构与实物相似,但视觉上与实际物体存在较大差距。为提高虚拟监控系统的真实性,需对构建的三维模型进行处理,三维模型的处理一般在3DMax进行,主要包括给模型添加材质、贴图等操作,使三维模型具备物理属性和纹理色彩,最终实现处理后的三维模型从物理属性、纹理色彩、光线等与现实世界中的物体尽可能一样。

3、三维场景构建

本文选择Unity3D作为虚拟车间三维监控系统的开发平台,场景是Unity3D程序的基本组成单位,场景管理主要是对场景中的模型进行组织和渲染。由于生产现场环境复杂,设备种类多样,造成虚拟监控系统中的模型数量巨大,

加大了场景管理的难度。场景的组织方式对系统后期设备的运动控制、环境渲染、三维漫游和数据源的数量和接口方式都有重要影响。本文提出采用场景树的方式实现场景的有效管理,一个场景树包含虚拟场景中的所有信息,既包括树的顶点信息,也包括对象变换和渲染等信息,许多不同类型的节点按照层次结构进行排列,最终组成场景树。通过场景树的方式大大降低了场景管理的复杂度,为系统开发提供了便捷方法。

4、运动控制技术

运动控制技术是实现虚拟监控技术的核心。在对虚拟模型运动控制的基础上,用生产现场的实时数据驱动模型,最终实现所有运动的虚实同步。常用的运动控制包括基本运动控制、搬运设备运动控制和机械手运动控制等。

(1) 基本运动控制。

由场景树可知,场景中的对象之间存在一定的层级关系,在Unity3D中子节点跟随父节点进行运动,故当一个对象发生运动时,组成它的所有子节点、子子节点等都将一起发生运行。当脚本写好后,通过拖拽方式将脚本挂在相应的对象上,就可实现该对象及其所有子对象的运动控制。

Unity3D最基本的运动是对象的移动和旋转,通过节点层级关系、特殊插件、几何关系计算等可实现复杂对象的复杂动作。对象移动的实现代码如下:

```

Void ObjectMovement()
{
    Transform.Translate(position, relative);
}
  
```

实现对象在relative坐标系中移动position距离。position可为Vector3类型或float类型的一维、二维、三维值,表示对象相对于relative坐标系沿x、y、z轴的移动量。

对象旋转的实现代码如下:

```

Void ObjectRotation()
{
    Transform.Rotate(angle, relative);
}
  
```

实现对象沿relative坐标系旋转angle角度。angle可为Vector3类型或float类型的一维、二维、三维值,表示对象相对于relative坐标系沿x、y、z轴的旋转量。

(2) 搬运设备运动控制。

搬运设备运动控制可通过基本运动控制中的移动和旋转代码来实现,但此方法没有充分利用生产车间搬运设备路径

相对固定的优点，运动控制效率低且需要不断获取搬运设备的位置信息和转动信息，对数据的连续性、实时性要求极大，加大了系统的运行开销，容易造成系统卡顿问题。

DOTween 是 Unity3D 中最常用的插件之一，在动画制作方面具有令人折服的能力，本文提出采用 DOTween 插件实现搬运设备的运动控制。实现节点之间的路径运动是 DOTween 插件中的路径编辑器 DOTween Path 组件。

```
Using DG.Tweening;
public class dotweening : MonoBehaviour
{
    void dotweenMove()
    {
        Transform.DOPath(Vector3[] waypoints, float duration,
            PathType pathType, PathMode pathMode, int resolution,
            Color gizmoColor);
    }
}
```

waypoints 是运动路径上各个节点的位置坐标，可以为数组，同时包括多个节点；duration 是相邻节点之间的运动时间；pathType 是运动路径的类型，包括 Linear（直线）和 CatmullRom（弯曲）两种；pathMode 是路径模式，用于选择正确的 LookAt 选项，包括 Ignore（不动）、3D（三维模式）2D（二维模式）三种模式；resolution 是路径的分辨率，值越大越适合长弯曲路径，默认值为 10；gizmoColor 是路径的颜色，当程序运行时路径显示的颜色，默认是白色。

根据生产车间实际搬运设备的路线，使用 DOTween 插件在虚拟监控系统的对应线路上添加一定数量的节点，根据添加顺序，节点将自动相联构成一条线路，设计的线路相当于搬运设备的路径约束，搬运设备会沿着这条路线进行运动。每个节点的位置坐标可采用脚本进行修改，即实现了搬运路径的约束性也实现了搬运路径的动态性。这样可实现当搬运设备间断地发送位置信息时，虚拟监控系统中的搬运设备仍能沿着一定的路径和速度平滑的进行运动，既降低了系统开销也达到了良好的展示效果。

（3）机械手运动控制。

机械手运动控制中各个关节之间的运动约束非常复杂，采用基本运动控制中的移动和旋转代码很难保证机器人运动的协调性和流畅性。本文提出机械手的运动控制采用添加 IK 插件的方式实现。IK（Inverse kinematics）即反向运动学，用子节点带动父节点的运动，常见的 IK 插件包括 Aim IK、Biped IK、CCD IK、FABRIK 等模块，每种 IK 的应

用场景不同，其中 CCD IK 是通过不断迭代运算求出一个近似解，来决定各层级父子关节的姿态，求解效率高、效果好，故其应用较多。

给机械手模型添加 IK 插件，实现通过控制机械手末端位置来保证整个机械手整体协调运动的目的，以 CCD IK 为例，其实现流程如图 2 所示。

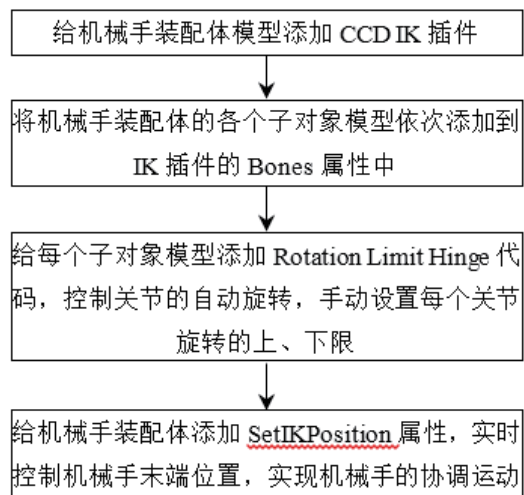


图 2 机械手运动控制实现流程

控制关节运动的 Rotation Limit Hinge 代码是系统自带的，用户只需根据机械手运动的实际情况，手动调整每个关节可旋转角度的上下限即可，在此不再累述。控制机械手末端的位置，实现整个机械手协调运动，实现代码如下：

```
public class SetIkPosition : MonoBehaviour
{
    public Transform targetToSync;
    CCDIK ccd;
    void Start ()
    {
        ccd = GetComponent<CCDIK>();
    }
    void LateUpdate ()
    {
        ccd.solver.IKPosition= targetToSync.
        position;
    }
}
```

5、人机交互技术

人机交互技术是指通过外部硬件输入，系统能做出相应的响应，最终实现人与计算机的信息交互，其中鼠标和键盘是当今最常用的人机交互输入设备。通过人机交互技术实现虚拟车间三维监控系统的人机交互，不仅能更好地展示生产

数据，还能提高人的参与度，使监控效果趋于智能化。虚拟车间三维监控系统人机交互主要是设备信息动态查看和三维场景切换两个方面，本系统也将鼠标和键盘作为输入设备。

(1) 设备信息查看。

虚拟车间三维监控系统中所有设备实时状态信息最好的查看方式是通过鼠标点击相应设备模型弹出相应对话框进行显示，因为这种方式简单、直观，符合人的主观认知。要实现通过鼠标点击实现设备信息的查看，首先需要虚拟监控系统能够接收鼠标点击事件，其次通过一定的方法得到鼠标点击的设备，然后读取该设备的状态信息，最后弹出设备状态信息窗口。

虚拟车间三维监控系统的鼠标交互操作是通过 Unity3D 中的输入接口函数 Input 来实现，KeyCode.Mouse0 代表鼠标左键，通过 Input.GetKeyDown(KeyCode.Mouse0) 实现外部鼠标点击事件的输入，将实现过程放入脚本自带的 Update() 函数中，保证实时响应用户的操作。

(2) 三维场景切换。

人机交互模块的三维场景切换主要包括关键工位三维场景的切换、生产现场多视角切换、生产现场指定路径漫游和人员控制漫游路径观察四部分。这四部分涉及的技术基本相同，都是通过鼠标或键盘的输入技术和相机的移动和旋转来实现。

相机的移动、旋转、位置和角度控制是三维场景切换效果的重要影响因素。相机的运动和旋转操作的原理和其他对象一样，通过 Transform 类的 Translate 和 Rotate 函数来实现。可在一个场景中放置多个相机，实现在界面上同时观察多个位置的设备运行情况，与生活中车库等监控系统的多视角同时观察、监控效果相似。生产现场按指定路线进行漫游可通过 DOTween 插件实现，根据实现约定好的漫游路径，在 DOTween 中拖拽式的画出对应的漫游路径，当需要按指定路径进行漫游时，让相机按照 DOTween 规划的路径进行运动即可，实现方式简单、漫游效果好。人员控制漫游路径进行观察通过键盘或鼠标的输入，调用 Transform 类的 Translate 和 Rotate 函数来实现相机为移动和旋转，最终实现人员控制漫游路径，可从任何位置 and 任何角度观察生产现场，达到良好的人机交互效果。

6、三维场景与网页数据交互技术

三维场景与网页的数据交互技术主要涉及数据格式和两

者的交互方式两方面问题。数据格式对数据交互的效率、速度、解析的难易程度都有重要影响，三维场景与网页的数据交互是实现生产数据驱动模型运动和人机交互的基础。

(1) 数据格式。

数据格式是三维可视化插件与网页进行数据交互首先要考虑的问题，但由于网页与三维可视化插件在数据接口、应用环境等方面存在较大差异，数据格式的定义是两者进行数据交互的难点。结合网页与插件的传输接口、各种数据格式的特点，最终选用传输方便、兼容性强、直观性强、便于解析优点的 JSON 字符串作为三维可视化插件与网页数据交互的格式。

(2) 函数调用。

三维可视化插件属于网页的嵌入模块，作为一个相对独立的模块，要想实现两者的数据交互需要一个通道，为实现三维可视化插件与网页的双向响应，通道要实现两者数据和函数调用的双向性。本文提出采用函数调用的方式实现三维可视化插件与网页的数据交互。

三维可视化插件调用网页中的函数。三维可视化插件调用网页中的函数通道已经被封装在 Application 类，可使用 Application.ExternalCall (functionName, args) 调用网页中的函数（只用于 Web Player），参数 functionName 表示网页中的函数名，参数 args 表示传递的参数，支持原始的数据类型（string、int、float、char）和 JSON 字符串格式。这个函数调用在程序执行时不会被阻塞，即 ExternalCall 立即被返回而不必等待被完成，且传递的参数数量是可变的。

网页调用三维可视化插件中的脚本函数。网页调用三维插件中的脚本函数是通过 SendMessage() 函数实现，函数原型为 SendMessage(Object, functionName, params)，参数 Object 表示三维插件中的 GameObject 模型对象，参数 functionName 表示挂载在 GameObject 对象的脚本函数名，参数 params 表示传递的参数，支持原始的数据类型（string、int、float、char）和 JSON 字符串格式。在使用 SendMessage() 函数进行调用之前，必须首先使用 JavaScript 中对象 Document 的 getElementById() 函数来获得 Unity Web player 的引用。

通过 Application.ExternalCall() 和 getUnity().SendMessage() 函数以及以上的相关规则可实现三维可视化插件与网页之间函数的双向调用，实现三维场景与网页的数据交互。

三、虚拟车间三维监控系统设计

1、系统架构设计

虚拟车间三维监控系统基于 B/S 架构，通过实时通信服务器、历史数据库等数据驱动虚拟监控系统中的三维模型，在 Unity3D 平台上实现三维模型与生产现场设备、物料、环境和生产进度等信息的实时同步，并保证用户通过人机交互界面实现对设备、具体工位的查看和场景漫游功能，系统的架构如图 3 所示。

虚拟车间三维监控系统的架构分为五层。

(1) 物理层。物理层主要是指生产现场硬件层和数据采集系统两部分，物理层主要负责产品的生产和装配，也是所有生产信息的源头。数据采集硬件将实时采集生产数据，并通过传输网络实时上传生产信息。

(2) 数据层。数据是虚拟车间三维监控系统的动力。虚拟车间三维监控系统中模型的实时驱动、生产状态信息、搬运信息、警报信息等都需要生产现场的实时数据、历史数据和控制指令信息。通过函数调用方式以 JSON 数据格式完成虚拟场景与数据层数据的实时交互，驱动虚拟车间三维监控系统的运动，实现虚实同步。

(3) 业务逻辑层。业务逻辑层根据生产现状对虚拟车间三维监控系统要展示的信息进行设计，主要包括加工设备信息模块、物流设备信息模块、加工进度信息模块和报警信息模块四个方面。

(4) 功能模块层。功能模块层主要是对虚拟车间三维监控系统人机交互功能进行设计，包括生产车间多视角切换、设备详细信息查看、生产现场多视图展示和场景漫游四个模

块。通过这四个功能模块的设计，提高可视化系统的交互化控制和展示效果。

(5) 界面层。界面层是虚拟车间三维监控系统的最终展示层，将生产现场的所有信息进行完全虚拟化展示，实现虚拟监控系统与生产现场实时同步。

2、功能模块设计

结合系统展示的内容和方式，虚拟车间三维监控系统要实现生产车间的多视角切换、设备详细信息查看、生产现场多视图展示和场景漫游四个功能模块。

(1) 生产车间的多视角切换。

虚拟车间三维监控系统要实现生产车间的多视角切换功能。因为生产现场物理空间较大、生产设备多、加工流程多等因素，从一个固定角度很难观察整个生产过程，为保证系统展示的全面性、完整性和效果最佳化，系统需实现从多视角观察整个虚拟车间的生产情况。这些视角包括系统的多个全景视角和关键工位、关键设备的局部视角，工作人员通过点击视角切换按钮实现监控系统的视角转换，从多方向、多角度观察与生产现场同步的虚拟车间三维监控系统，达到了展示效果的最佳化。

(2) 设备详细信息查看。

虚拟车间三维监控系统运行时，虚拟场景中的设备会与生产现场的加工设备同步运动，主要展示的是加工过程的运动信息，而设备的详细加工参数信息不会直接在监控系统界面上展示出来。系统需实现当用户用鼠标点击虚拟场景中的任意一台设备时，弹出对话框详细显示此台设备的运行数据、基础数据等信息。

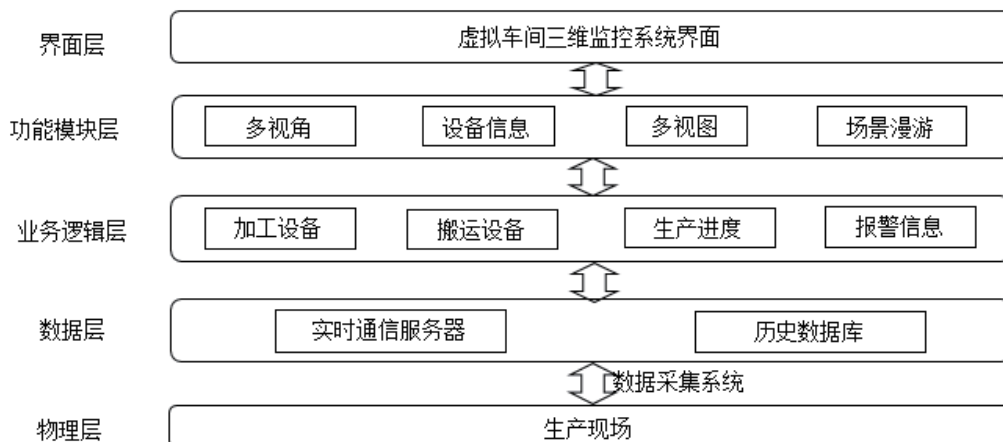


图 3 虚拟车间三维监控系统架构

(3) 生产现场多视图展示。

生产现场多视图展示是指三维监控系统界面会同时显示多个视图界面。生产过程中生产关键信息较多，一个视角很难同时展示多个重要的实时信息，故本系统需实现同时显示多个视图的界面，达到多视图、多信息同时展示的最佳效果。

(4) 场景漫游。

视角展示和设备信息展示都是以固定的视角对虚拟场景进行展示，场景漫游可实现动态、多视角、移动地展示虚拟场景的实时运行信息。本系统需实现的场景漫游包括按指定路线进行漫游和人工操作进行漫游两种方式。通过场景漫游可深入虚拟车间三维监控系统的内部进行展示，与人员进入车间现场进行观察的效果相似，提高了展示的真实性、形象性和交互性。

四、虚拟车间三维监控系统实现

本文对虚拟车间三维监控系统的实现流程进行了详细的分析和研究，实现了系统的场景搭建、运动控制、人机交互以及虚拟监控系统与现场生产数据的交互，但当数据进入虚

拟系统后，怎样对庞大、复杂、冗余的数据进行解析、传输并驱动虚拟场景中对应模型的运动仍是虚拟监控系统实现的难点。此处将从系统开发的角度以数据流向为导引，详细讲解虚拟车间三维监控系统内部数据解析、传输与模型驱动的实现方法。

虚拟车间三维监控系统数据获取采用按一定频率主动读取的方式，通过 Unity3D 提供的 UnityWebRequest() 接口采集系统外部 JSON 格式生产数据，系统将数据转换成 JsonData 格式，然后对数据进行解析处理，将数据根据工位进行分解、提取并在提取的数据前端添加对应工位的工位名。系统根据工位名和相应请求构成的字符串字典将相应的数据传输给对应工位的控制类，控制类将数据重新转换成 JSON 格式，给工位的各个变量进行赋值，通过数据驱动模型运动，系统内部的数据流逻辑如图 4 所示。

虚拟车间三维监控系统以工位为单位，每个工位模型都有对应的数据类和运动驱动类，数据类以一定的频率发出数据请求指令，最后将获取的数据传输到运动驱动类，运动驱动类通过数据类发来的实时数据，驱动虚拟模型的运动。

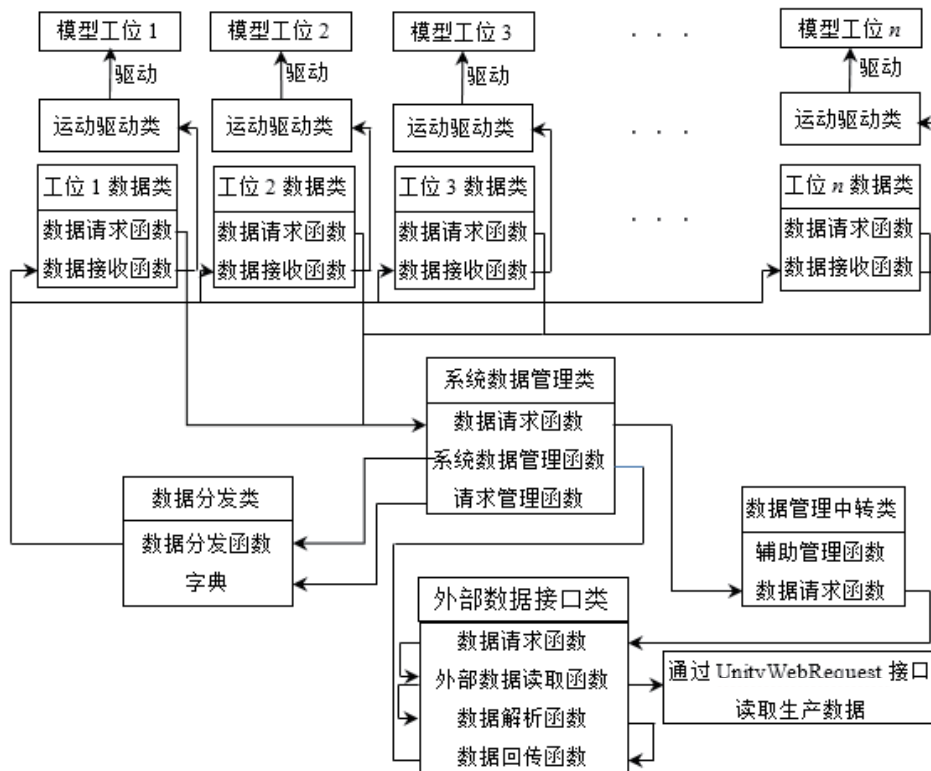


图 4 虚拟车间三维监控系统数据流逻辑图

外部数据接口类中外部数据获取函数的实现代码如下：

```
IEnumerator GetJson(string requestURL)
{
    using (UnityWebRequest www = UnityWebRequest.
Get(requestURL))
    {
        yield return www.Send();
        if (www.isError)
        {
            Debug.Log(www.error);
        }
        else
        {
            string data = www.
downloadHandler.text;
            DealDataAndpass(receivedMsg,
ProcessDataCallback);
        }
    }
}
```

数据分发类中的数据分发函数实现代码如下：

```
public void HandleResponse(StationName stationName,
LitJson.JsonData jsonData)
{
    IRequest request=null;
    stationRequestDic.TryGetValue(stationName, out request);
    if (request == null)
    {
        Debugger.Log(“ 获取失败! ”);
        return;
    }
    else
    {
        Debugger.Log(“ 获取成功! ”);
    }
}
```

```
request.OnResponse(jsonData);
}
```

以上对系统内部数据流转关键节点的实现过程进行了详细阐述，加之其他辅助函数实现虚拟车间三维监控系统内部数据流的畅通，保证了系统的顺利实现。

五、案例分析

以车载雷达电子装配线为对象，验证本文提出的虚拟车间三维监控系统实现流程的合理性、可行性和实用性。通过三维建模、模型处理、三维场景构建、运动控制、人机交互、三维场景与生产数据交互、数据解析与传输等技术开发了车载雷达电子装配线虚拟车间三维监控系统。系统实现了生产车间的多视角切换、设备详细信息查看、生产现场多视图展示和场景漫游功能。车载雷达电子装配线虚拟车间三维监控系统如图 5 所示。

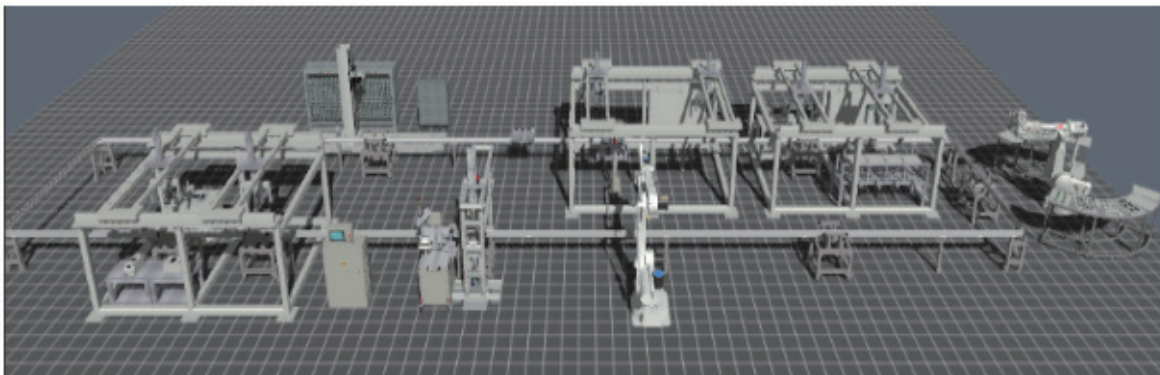


图 5 车载雷达电子装配线虚拟车间三维监控系统