李广松, 杨艳瑜, 吴绍棋. 基于 HTC 手势识别的计算机组装 VR 交互系统设计[J]. 智能计算机与应用,2025,15(9):153-160. DOI:10.20169/j.issn.2095-2163.250924

基于 HTC 手势识别的计算机组装 VR 交互系统设计

李广松, 杨艳瑜, 吴绍棋
(广东职业技术学院, 广东 佛山 528041)

摘 要: 采用 Unity 3D 引擎、SteamVR 插件及 Vive 手部追踪 SDK,研发出一款兼具传统 VR 交互和手势交互的计算机组装教学系统。该系统严格遵循笔记本和台式机硬件标准,运用 3DS MAX 构建虚拟模型,确保高度逼真。借助 SteamVR 与 Unity3D 的集成,实现基础 VR 交互;同时,结合 Vive 技术与 HTC Vive 设备,实现精准手势识别。在 Windows 10 环境下经过全面测试,证实系统稳定且功能完善,满足教学需求。该研究为计算机组装教学带来技术创新,并为手势交互与 VR 结合提供实践范例。

关键词: HTC; 虚拟现实; Unity 3D; 计算机组装
中图分类号: TP391 **文献标志码**: A **文章编号**: 2095-2163(2025)09-0153-08

Design of computer assembly VR interaction system based on HTC gesture recognition

LI Guangsong, YANG Yanyu, WU Shaoqi
(Guangdong Polytechnic, Foshan 528041, Guangdong, China)

Abstract: A computer assembly teaching system is developed that combines traditional VR interaction and gesture interaction using Unity 3D engine, SteamVR plugin, and Vive hand tracking SDK. The system strictly follows the hardware standards of laptops and desktops, and uses 3DS MAX to build virtual models, ensuring high realism. The integration of SteamVR and Unity3D is utilized to achieve basic VR interaction; meanwhile, by combining Vive technology with HTC Vive devices, precise gesture recognition can be achieved. After comprehensive testing in the Windows 10 environment, it has been confirmed that the system is stable and fully functional, meeting teaching needs. This study brings technological innovation to computer assembly teaching and provides practical examples for the combination of gesture interaction and VR.

Key words: HTC; virtual reality; Unity 3D; computer assembly

0 引 言

传统计算机组装课程长期以来一直面临着诸多挑战[1-2]。高昂的训练设备费用、硬件资源的高损坏率,以及电子元器件可能引发的火灾风险,都是不容忽视的问题。然而,借助虚拟现实(VR)技术[3],可以有效地规避这些难题。通过 VR 虚拟场景进行课程训练[4-7],不仅能显著降低对实体资源的依赖,进而大幅减少课程成本,使更多学生受益,而且在操作过程中即使出现失误,也不会像实体机器那样带来安全隐患。更重要的是,虚拟资源的更新速度快,可以紧跟市场步伐,模拟最前沿的硬件设备。

尽管如此,现有的 VR 实训方案[8-9]也并非完美无缺。虚拟环境本身的局限性导致了视觉和触觉体验的不佳,以及场景真实感的缺失。在解决虚拟现实的体验感问题上,国内外的学者进行了大量研究。其中,张梓刚[10]提出通过 HTC VIVE 设备[11-12]的触摸板进行文本输入,利用双手操作,大大提高了输入效率。刘磊[13]通过录制手势样本,利用 VGB 和 Adaboost 算法进行训练,实现了 PC 端的手势交互。林莹莹等学者[14]借助 LeapMotion 设备,开发出基于 HTC 的裸手交互方式,进一步摆脱了对外部设备的依赖。张琦[15]提出将手部图像与虚拟现实相融合的方法,减少对手柄的依赖,提升了用户的沉浸感。

基金项目: 广东省 2022 年科技创新战略专项资金("攀登计划"专项资金)项目(pdjh2022a0825)。
作者简介: 杨艳瑜(1993—),女,助教,主要研究方向:虚拟现实,教育技术研究。
通信作者: 李广松(1980—),男,副教授,主要研究方向:游戏开发,XR 技术。Email: 41833383@qq.com。
收稿日期: 2024-01-03

本文针对传统计算机组装 VR 交互的问题,提出使用 HTC Vive 设备和 Vive Hand Tracking SDK 的解决方案。该方案有效解决了用户体验不真实、运动自由度受限以及手部疲劳等问题,显著提升了用户体验的真实感和自由度。

1 系统架构

HTC Vive 是一款在虚拟现实领域中备受推崇的经典头戴设备。其核心技术在于通过精密的定位器和内置的摄像头,实现了对用户位置的高精度追踪以及实时画面的捕捉,从而为用户带来了沉浸式的虚拟现实体验。在构建虚拟现实交互系统时,系统选择了 Unity 作为开发平台。Unity[16-17]作为一款功能强大的 3D 引擎,不仅在游戏行业[18-20]中得到了广泛应用,也在虚拟现实领域[21-22]展现了其卓越的性能。Unity 还提供了丰富的工具和资源库,支持跨平台开发,使得开发者能够高效、便捷地创建出具有高度真实感和交互性的虚拟现实环境。

为了将 HTC Vive 与 Unity 完美结合,本文提出的基于 HTC 的计算机组装 VR 系统架构如图 1 所示。由图 1 可知,系统采用了 SteamVR Plugin 作为桥梁,实现了对头戴显示器、控制器等设备的无缝对接。同时,系统以 SteamVR Plugin 的交互系统为基础,进行了深度的定制和拓展,以满足更为复杂和个性化的交互需求。此外,系统还引入了 Vive Hand Tracking SDK,这一技术能够实时获取 HTC Vive 设备上捕捉的画面,对用户的手部动作进行精确追踪和识别。系统根据这些识别结果判定用户行为,并将其转发到交互系统中,从而实现了自然、直观的手势交互功能。这一创新性的技术应用,很好地提升了用户在虚拟环境中的沉浸感和操作体验。

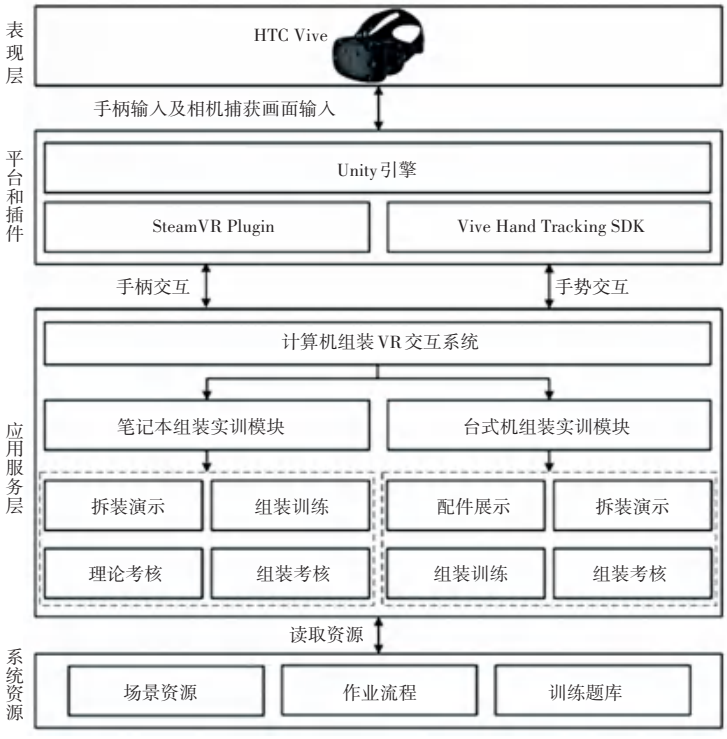

图 1 基于 HTC 的计算机组装 VR 系统架构

Fig. 1 Architecture of computer assembly VR system based on HTC

2 设计流程

本设计专注于在 Windows 系统上的 SteamVR 应用平台中集成手势交互的虚拟现实应用。基于 Unity 游戏引擎的强大功能,团队研究并实现了与 SteamVR 的紧密结合,特别是通过引入 Vive Hand Tracking SDK,为 VR 应用增添了直观且高效的手势交互能力。整个开发流程包括在 Unity Editor 中使用 Package Manager 导入 SteamVR Plugin 2.0,并进行完整的 VR 配置,随后整合项目所需的各类素材和预设动画资源。系统精心配置了 VR 交互组件,并编写了控制模型、UI 和动画的脚本,确保交互流程的顺畅与自然。此外,团队对 SteamVR Plugin 进行了拓展,成功接入了 Vive Hand Tracking SDK,实现了手势与 VR 环境的无缝对接。最终,经过细致的优化、调试和测试,将应用打包为. exe 格式,确保

在配备 HTC Vive 设备和 SteamVR 的 Windows 10 系统上能够流畅运行。

3　项目搭建

3.1　配置 VR 项目

通过 UnityHub 以 3D 模板创建 Unity 项目。登录 Unity Asset Store，搜索并找到 SteamVR Plugin，并将其添加到个人资源中。在 Unity 的 Package Manager 窗口中选中 My Assets 并找到 SteamVR Plugin，点击右下角 Import 导入到过程中。为了验证 SteamVR Plugin 是否正确安装并能够正常工作，建议打开 Assets 文件夹中的 SteamVR \ InteractionSystem \ Sample 目录，并运行其中的 Interactions_Example 场景。如果所有示例互动都能正常运行，那么就说明插件已正确安装并可以正常使用。

需要注意的是，运行时本机环境应该具备 SteamVR 软件，并确保 OpenVR 成功安装到 Unity 工程中。

3.2　导入 Vive Hand Tracking SDK

为了在 Unity 中配置和导入 Vive Hand Tracking SDK，请按照以下步骤进行操作：打开 Unity，并选择“Edit”菜单中的“Project Settings”选项。在“Project Settings”窗口中，选择“Package Manager”设置项。在“Package Manager”设置中，点击“Add”按钮，并选择“Scope Registry”选项。在“Scope Registry”窗口中，填写以下配置信息：Name：Vive，URL：https://npm-registry.vive.com/，Scope(s)：com.htc.upm。点击“Add”按钮，成功添加作用域注册表。关闭“Project Settings”窗口，并选择“Window”菜单中的“Package Manager”选项。在“Package Manager”窗口中，选择“My Registries”选项卡。在“My Registries”选项卡中，找到 Vive Hand Tracking SDK。右键单击 Vive Hand Tracking SDK，并选择“Install”选项。等待安装完成。

请注意，具体的安装方法可能会因 SDK 版本而有所不同，建议参考官方文档进行操作。另外，添加作用域的方式仅在 Unity 2019 及以上版本支持。

4　系统功能实现

4.1　接入 Vive Hand Tracking SDK

在 SteamVR 应用程序中，输入内容的处理流程是关键。首先，通过 SteamVR 软件平台获取 HTC Vive 设备的输入内容，这一信息经过封装后转发到 OpenVR 中。因此，只要接入了 OpenVR 的程序，就能获取到这些信息。在 Unity 的 SteamVR Plugin 中，同样也是通过 Unity 导入 OpenVR 后获取输入内容。这个插件还专门针对 OpenVR 封装了一个输入系统，其设计强大，能够兼容其他类似规范的输入。为了将 Vive Hand Tracking SDK 的输入内容与 SteamVR Plugin 的输入系统相匹配，还需要进行一系列的设置和调整。

首先，在 SteamVR_Input_Sources 中新增一个自定义的枚举类型，比如 GestureLeftHand。这样能够更好地组织和识别不同的手势。接着，为 SteamVR_Action_Source 添加一个能够设置回调处理的入口。例如，创建一个字典 customUpdateValue，用于储存对应输入源、对应行为的回调。这样可以在需要时快速地调用相应的处理函数。

在 Awake 或 Start 时，给 customUpdateValue 传入新增类型的处理函数。同时修改或新增一个 SteamVR_Action_Source 的实现类，比如 SteamVR_Action_Boolean_Source。在这个实现类中，将 UpdateValue 阶段的输入源获取改为回调获取。这样在每个 UpdateValue 调用时，都会通过回调返回 GestureProvider 的响应结果。

需要注意的是，以上仅为一个示例方案，具体的实现应考虑拓展性和兼容性问题。在实际应用中，可能还需要根据 SDK 的具体文档和要求进行适当的调整和优化。示例代码如下：

```
//为 SteamVR_Input_Sources 添加枚举
[Description("/user/gesturehand/left")]
GestureLeftHand,
//customUpdateValue 字典结构示例
Dictionary<SteamVR_Input_Sources, Dictionary<string, RefFunc<bool, object>>> customUpdateValue
//SteamVR_Action_Boolean_Source.UpdateValue 阶段
//如果用户更新事件不为空，则使用用户自定义更新
bool customIsBlock = false;
if (customUpdateValue.ContainsKey(inputSource)&&customUpdateValue[inputSource].ContainsKey(fullPath))
{
object o = customUpdateValue[inputSource][fullPath](ref customIsBlock);
actionData = (InputDigitalActionData_t)o;
}
```

```
if (customIsBlock == false)
{
    EVRInputError err = OpenVR.Input.GetDigitalActionData(action.handle, ref actionData, actionData_size, inputSourceHandle);
    if (err != EVRInputError.None)
    Debug.LogError("<b>[SteamVR]</b> GetDigitalActionData error (" + action.fullPath + "):" + err.ToString() + " handle: " + action.handle.ToString());
}
//传入手势识别结果回调
SteamVR_Input_Sources input = isLeft ? SteamVR_Input_Sources.GestureLeftHand : SteamVR_Input_Sources.GestureRightHand;
AddActionUpdate(input, action, (ref bool v) =>
{
    GestureResult result = isLeft ? GestureProvider.LeftHand : GestureProvider.RightHand;
    v = true;
    InputDigitalActionData_t data = new InputDigitalActionData_t();
    data.bActive = true;
    data.bState = result.IsFlagMatch(handFlag);
    data.bChanged = (result.IsFlagMatchDown(handFlag)
    || result.IsFlagMatchUp(handFlag));
    return (object)data;
});
```

4.2 左右旋转场景

在 Unity 的 SteamVR Plugin 中,通过 SnapTurn 组件实现手柄的左右旋转功能。SnapTurn 组件是一个交互组件,允许玩家在 VR 环境中快速转身。这个组件已经具备了用户所需的大部分功能,只需要在此基础上调整转动的角度大小、速度、特效音效以及操作按键即可。然而,为了实现手势控制左右旋转的功能,需要进行一些额外的设置。由于 SnapTurn 组件并不支持输入源,也就是操作手柄的设置,需要添加一些额外的设置项。

首先,需要导入 SteamVR Plugin 和相关命名空间。然后,创建一个自定义的旋转控制器,例如 RotateController。在这个控制器中,可以设置手柄旋转的灵敏度和速度,并初始化 SnapTurn 组件和相关变量。在 Update 方法中,获取手柄输入并计算旋转角度。然后,将旋转角度应用到 SnapTurn 组件上。为了限制旋转角度的范围,使用 Mathf.Clamp 方法来确保角度值在合理的范围内。

通过这种方式,可以实现通过手势控制左右旋转的功能。需要注意的是,由于手势识别技术的限制,这种控制方式可能会有一定的可承受范围内的延迟和误差。示例代码如下:

```
//为组件提供设置入口
public SteamVR_Input_Sources snapLeftSources = SteamVR_Input_Sources.LeftHand;
public SteamVR_Input_Sources snapRightSources = SteamVR_Input_Sources.RightHand;
//更改组件的核心控制源码(位于 SnapTurn.Update()中)
bool leftHandTurnLeft = snapLeftAction.GetStateDown(snapLeftSources) && leftHandValid;
bool rightHandTurnLeft = snapLeftAction.GetStateDown(snapRightSources) && rightHandValid;
bool leftHandTurnRight = snapRightAction.GetStateDown(snapLeftSources) && leftHandValid;
bool rightHandTurnRight = snapRightAction.GetStateDown(snapRightSources) && rightHandValid;
//管理 SteamVR Pluin 中手势的输入
GestureTools.AddBoolAction("/actions/default/in/SnapTurnLeft", true, HandFlag.Like);
GestureTools.AddBoolAction("/actions/default/in/SnapTurnRight", false, HandFlag.Like);
public static void AddBoolAction(string action, bool isLeft, HandFlag handFlag)
{
    SteamVR_Input_Sources input = isLeft ? SteamVR_Input_Sources.GestureLeftHand : SteamVR_Input_Sources.GestureRightHand;
    AddActionUpdate(input, action, (ref bool v) =>
    {
        GestureResult result = isLeft ? GestureProvider.LeftHand : GestureProvider.RightHand;
        v = true;
        InputDigitalActionData_t data = new InputDigitalActionData_t();
        data.bActive = true;
        data.bState = result.IsFlagMatch(handFlag);
        data.bChanged = (result.IsFlagMatchDown(handFlag)
```

```
        || result.IsFlagMatchUp(handFlag));
    return (object)data;
        });
    }
```

4.3 组装实操模块

在 Unity 的 SteamVR Plugin 中，通过交互系统和程序逻辑来实现用户与硬件模型的交互，同时利用场景预设模型的高亮效果来提醒用户，并结合 UI 的文字介绍来完成整个教学训练过程。以笔记本组装为例，按照以下步骤进行操作：首先，准备一个完整的笔记本模型，确保其各个硬件模块都是独立的。然后，复制一份这个模型，并将所有模型渲染隐藏，只显示硬件模块。将这个复制的模型打乱摆放到场景中，作为未组装状态下的硬件模型副本。通过代码来控制这些硬件模块的位置，甚至模拟出一种混乱的效果。接下来，再复制一份笔记本模型，这次将除了固有内容之外的所有渲染都隐藏，创建出组装完成状态下的副本。而原始的笔记本模型则作为玩家实际操作的交互对象。

然后，给当前操作的副本模型添加 Interactable 交互组件和自定义的 NotebookAccessories 脚本。在 NotebookAccessories 脚本中，编写具体的交互逻辑和记录硬件信息，包括对未组装和组装完成状态下的副本引用的管理。最后，编写一个 NotebookManager 脚本，用于管理所有笔记本组件以及程序控制。这个脚本可以实现各种功能，如重置操作、组装进度的检测以及教学步骤的提示等。

通过这种方式，可以在 VR 环境中提供一种沉浸式的笔记本组装体验，同时通过交互和 UI 指导用户完成整个过程。这种教学方法可以大大提高用户的参与度和学习效果。示例代码如下：

```
//笔记本配件实例类
[RequireComponent(typeof(Interactable))]
public class NotebookAccessories: MonoBehaviour
{
    private string name;
    private string description;
    public Sprite descriptsImage;
    private Transform originPositionTran;
    //跟随事件
    private void OnAttachedToHand(Hand hand)
    {
      SetAssemblyHint(true);
      SetAccessoriesName(false);
      if(descriptsImage ! = null)
       NotebookManager.instance.SetAccessories Descripts
(descriptsImage);
    }
    //检测是否达到装配点
    private void OnDetachedFromHand(Hand hand)
    {
      SetAssemblyHint(false);
        //如果已经抵达位置，则开始检测装配
      if (isArriveAssemblyPosition)
      {
            //顺序正则放入
            if (Test())
            {
              //显示错误提示，复位当前配件
                NotebookManager. instance.
                ShowErrorTip();
                ResetPosition();
            }
      }
    }
}
```

4.4 考评模块

在实操模块的基础上，为了模拟真实的考试环境，系统提供可配置的组装步骤以及关闭提示的功能。在判断配件组装是否正确时，除了顺序和位置，还需要考虑力度因素。但为了简化处理，本文仅针对顺序和位置进行逻辑处理。

首先，创建一个列表，将配件模型按照正确的组装顺序排列。当用户的组装顺序与这个列表不一致时，即可判断为组装错误。其次，在组装完成状态下的模型中，预设每个硬件的精确位置。当用户尝试组装时，系统会实时检测配件的位置和旋转值与预设值的相似度。如果相似度过低，说明该安装不标准。最后，通过 NotebookAccessories 中配置的分值信息，逐一比较用户的操作与标准操作的差异。任何不标准的操作都将按比例扣除相应的分值。通过这种方式，不仅能评估用户的组装顺序，还能对其组装位置的精确度进行评估，从而提供一个全面而准确的模拟考试体验。实例代码如下：

```
public class NotebookManager: MonoBehaviour
{
  public static NotebookGameManager instance;
    //动画演示模块的动画控制
```

```
        public void ExeAutoAssembly(bool isAssembly)
        {
          assemblyAnim[assemblyAnimationName].
speed = isAssembly ? -1f : 1f;
          if (isAssembly)
           assemblyAnim[assemblyAnimationName].time =
assemblyAnim[assemblyAnimationName].length;
          assemblyAnim.Play(assemblyAnimationName);
        }
        //开始评测系统
        private IEnumerator DoTest(Action onCompleted)
        {
          DoTestInit();
          yield return DoTestChocie();
          yield return DoTestPart2();
          yield return DoTestSettlement();
          if (onCompleted != null)
           onCompleted();
        }
        //初始化评测系统
        private void DoTestInit()
        {
          if (testAnswerInstances == null)
          {
            testAnswerInstances = new Dictionary<
string, GameObject>();
           foreach (var item in testAnswerPrefab)
           {
             GameObject go = Instantiate(item);
             item.SetActive(false);
             testAnswerInstances.Add(item.name, go);
           }
          }
        testChocieResult.Clear();
        }
        //评测系统-实操考评内容
        private IEnumerator DoTestPart2()
        {
            //复位所有配件
            NotebookAccessories[]
testNotebookAccessories = testHostAnim.transform.
GetComponentsInChildren<NotebookAccessories>();
            foreach (var item in testNotebookAccessories)
item.ResetPosition();
            //展开笔记本动画并等待动画结束
            testHostAnim["NoteBookShouChai"].speed =
2;
            testHostAnim.Play("NoteBookShouChai");
            while (testHostAnim.isPlaying == true)
yield return null;
            //初始化所有配件
            foreach (var item in testNotebookAccessories)
            item.OnExerciseStart();
              //接受 NotebookAccessories 中的组装
报告,用 list 存储组装顺序
             testThisAssemblyList.Clear();
             while (testThisAssemblyList.Count< test
NotebookAccessories.Length)
            yield return null;
          }
      }
```

5 导出与测试

5.1 导出设置

在准备导出可执行文件(exe)之前,需要进行一些关键设置。首先,在 Unity 的“Player Settings”中,选择“XR Plug-in Management”,然后在“Plug-in Providers”下拉菜单中选择“OpenVR Loader”选项。此外,导出时有几个注意事项。确保路径中不包含中文,以避免潜在的兼容性问题。同时,避免将导出的文件放在工程目录下,以减少可能出现的意外问题。这些预防措施有助于确保导出的程序稳定可靠。

5.2 调试和测试

构建完成后,进入测试阶段。首先,请确保已安装 Steam 客户端并完成登录。随后,启动 SteamVR,并连接好 VR 头戴式显示器和手柄设备。确保一切正常后,双击导出的应用程序文件,启动应用程序。戴上头戴式显示器,将进入 VR 场景。若希望在 Unity 中进行 Debug 测试,也需配置好 OpenVR 并启动 SteamVR。随后,像普通项目一样直接点击运行即可。此时,戴上头显,便能看到 VR 场景内容。同时,通过 Debug.Log 可以观察到程序运行过程中的日志信息,有助于定位问题。

完成 Debug 测试无误后,再进行打包可以大大缩短测试周期。打包过程中可能会遇到资源问题,需要较长时间处理。请注意,SteamVR 的性能表现依赖于用户的计算机性能。如果运行不够流畅,可

能需要对场景模型或 Draw Call 进行优化。当然，也可以直接在 Unity 中进行整体质量的调整。

5.3　测试结果分析

本文所描述的程序在 Intel 酷睿 i5 8350U+GTX 1650 的台式计算机、Windows10 系统上进行了全面测试。测试效果如图 2 所示。针对交互操作、手势识别，以及动画、语音、文字、特效和评测功能，测试结果显示：配件模型操作精准，手势识别准确率高于 95%，实时帧数稳定在 40 fps 以上，各类效果表现恰当且正常，评测功能结果准确。

(a) 台式机配件展示模块　(b) 笔记本拆装演示模块

(c) 台式机拆装演示模块　(d) 笔记本组装训练模块

(e) 台式机组装训练模块　(f) 笔记本理论考核模块

(g) 台式机组装考核模块　(h) 笔记本组装考核模块

图 2　测试效果图

Fig. 2　Diagram of test effect

6　结束语

系统基于 Unity 3D、SteamVR Plugin 和 Vive Hand Tracking SDK 进行开发，采用 C#编写交互脚本，设计并实现了一个兼具传统 VR 交互和手势交互的计算机组装教学系统。系统具备配件认识、组装练习、考核评测等功能。然而，目前系统优化不足，低性能计算机上实际运行效率不高，帧数不够高。后续研究工作将聚焦于性能优化，包括但不限于更换更高效的手势识别技术，以便进一步提升用户体验和适用性。

参考文献

[1] 钱武. 计算机组装与维护课程教学中存在的问题及对策研究[J]. 现代职业教育，2018(22)：57.

[2] 许力航. 计算机组装与维护课程教学中存在的问题及对策[J]. 辽宁高职学报，2011，13(1)：44-45.

[3] 李敏，韩丰. 虚拟现实技术综述[J]. 软件导刊，2010，9(6)：142-144.

[4] 姬志敏，文福安，谷文忠. 虚拟现实课堂五段式教学设计模型[J]. 数字教育，2020，6(2)：54-60.

[5] 程诚. VR 技术在计算机组装与维护中的应用[J]. 集成电路应

用,2023,40(12):112-113.
[6] 高倩. 基于沉浸式虚拟现实环境下虚拟实验的研究与设计:以《计算机硬件组装维护》课程为例[J]. 普洱学院学报,2021,37(3):35-37.
[7] 童严萍.《计算机硬件组装》虚拟实训系统的设计与应用[D]. 济南:山东师范大学,2021.
[8] 贾晓琪,芦囧耀. 基于虚拟现实技术的仿真教学平台的建设:以《计算机组装与维护》为例[J]. 智能计算机与应用,2023,13(10):88-91.
[9] 颜蕾. 基于虚拟仿真实训平台的情境教学模式构建与应用研究[D]. 大连:辽宁师范大学,2021.
[10]张梓刚. 虚拟现实环境下的新型双手文本输入方法研究[D]. 长春:吉林大学,2022.
[11] BRETON J, HOMOLA D. Lifeliqe partners with HTC vive to expand virtual reality into education[EB/OL]. (2016-07-14). http://caijing. chinadaily. com. cn/2016 - 07/14/content _ 26089720. htm.
[12]张琦. 基于 HTC Vive 的手势交互方法的应用研究[D]. 北京:北京工业大学,2020.
[13]刘磊. 自然交互技术在虚拟仿真教学中的应用与研究[D]. 武汉:江汉大学,2023.
[14]林莹莹,蔡睿凡,朱雨真,等. 基于 Leap Motion 的虚拟现实陶艺体验系统[J]. 图学学报,2020,41(1):57-65.
[15]张琦. 基于 HTC Vive 的手势交互方法的应用研究[D]. 北京:北京工业大学,2020.
[16]吴亚峰,于复兴,索依娜. Unity3D 游戏开发标准教程[M]. 北京:人民邮电出版社,2016.
[17]百纳科技. Unity 游戏案例开发大全[M]. 北京:人民邮电出版社,2015.
[18]刘贤梅,刘俊,贾迪. Unity 引擎下多人在线网络游戏的设计与开发[J]. 计算机系统应用,2020,29(5):103-109.
[19]周雪薇. 基于 Unity3D 的小学科学教育游戏的设计与开发[J]. 微型电脑应用,2021,37(1):44-46.
[20]史宝明,贺元香,李岚. Unity3D 随机寻路算法设计[J]. 吉林师范大学学报(自然科学版),2022,43(1):128-133.
[21]高天寒,李颖. 基于虚拟现实技术的汉语言教学系统设计与实现[J]. 工业和信息化教育,2018(8):79-83.
[22]张文胜,岳康. 基于 VR 的驾驶仿真及安全教育系统开发[J]. 计算机工程与应用,2022,58(23):278-284.