基于 AR 增强现实技术的校园社区交互

李翰韬 学号：16711094

（北京航空航天大学 自动化科学与电气工程学院, 北京 100191)

摘　　要：根据大学校园卡交互功能能够促进大学生建立校园社区这一现状，对校园社区的线下交互做出初步设想。建立通过线下 AR 增强现实扫描校园卡，以进入虚拟游戏或是虚拟社区，使当代大学生更好地融入集体生活。使用 Unity 与 Vuforia 建立简单的 AR 增强现实模型，使其与使用者可以进行简单的交互，交互方法包括使 3D 模型头部视角跟踪手机摄像头、使 3D 模型可以通过触摸位置点方式进行移动、为 3D 模型添加阴影等。后续工作中，只要将 3D 模型替换为自己设置的校园社区人物 3D 模型，即可建立相对完整的大学生校园社区。

关 键 词：Unity；Vuforia；AR 增强现实；校园社区；校园卡

1　引言

经过十多年的发展，融合了博客、BBS、邮箱、即时通讯等功能，通过每个人真实的人际关系，满足用户对资讯、娱乐、学习、社交等多方面需求的 SNS 社区，已经成为大学校园里最新的沟通方式、最流行的校园风尚，给相对单调、缺乏有效沟通的高校校园网络带来了生机。而如何将“无形”的高校校园网络文化渗透到“有形”的载体中，也就是以高校校园网络为依托，建立一个利于学校管理和服务的综合型社区，将大学日常管理、教学、科研和服务等方面渗透其中，营造一个利于学生成长、便于学校日常管理、善于形成正能量的校园网络环境，值得深入研究。

在传统的方式中，想要让使用者完全只在现实世界中相互交流，或者在交流中实现和现实世界的互动，都是很不方便的。虽然我们生活的世界是三维的，但我们的校园社区网络更偏重于二维的方式。 随着虚拟技术的发展，我们可以使用计算机生成的三维虚拟环境，通过建模给用户类似在真实世界一样的虚拟感受。但这些虚拟场景需要高性能的计算机图形支持，这使得构筑成本变得更高昂。尽管通过三维虚拟环境可以给学生提供很多沉浸感和兴趣点，但是也有一些隐患，比如用户完全沉浸在这种环境中时，他们就脱离了真实的环境。而且现在的虚拟技术水平也很难达到足够的现实水平。

一种新技术——增强现实（Augmented Reality，简称 AR）技术能够将物理世界和虚拟世界结合在一起，这为我们提供了新的思路。

2　案例分析

高校校园网络文化是以高校师生为活动主体、以高校校园文化为依托，通过网络进行信息沟通的行为方式及其道德和规范的总和，是校园文化和网络文化相互作用、相互影响的产物，已成为校园文化的重要组成部分。由于“高校校园网络文化”建设属于实践性比较强的问题，在现有研究中缺乏深入的实践性研究，更无法对实践产生指导作用。

而组建一个高校校园社区网络最重要的一点就是参与其中的人数。对此影响最为大的便是校园社区对学生的吸引力。拥有合适的用户间交流方法以及拥有吸引人的交流方式，便是吸引用户的重中之重。

日本的任天堂公司就曾经使用 AR 增强现实的方法丰富用户线下交互的手段，获得了很好的收益。日本任天堂公司在 2011 年发布了新的便携式游戏机 3DS（ 日文：ニンテンドー 3DS，英文：NINTENDO 3DS）。利用视差障壁技术，让玩家不需佩戴特殊眼镜即可感受到裸眼 3D 图像。在 3DS 的用户社交系统中，任天堂公司与以往不

完成日期：2019-01-02；
作者邮箱：hantao-li@outlook.com

同，提出了 AR 社交的方法。用户可以通过 AR 增强现实和摄像头，与自己创造出的虚拟人物进行交互，还可以通过 AR 增强现实进行简单的互动游戏。

图 1　3DS 中 AR 增强现实的介绍界面

Fig1　Introducing Interface of AR in 3DS

图 2　3DS 中自带的 AR 游戏

Fig2　The AR games in 3DS

除去在当时令玩家倍感新鲜的 AR 增强现实游戏外，任天堂还将用户的社区系统 Mii 整合进 AR 的整体系统内，让用户在体验 AR 游戏的同时，能够进入到社区内，与别的玩家一同交流。在线下游戏时，也支持了双人同时游戏的功能，大大吸引了当时的掌机玩家。

图 3　3DS 社区 Mii 在 AR 中的体现

Fig3　Embodiment of Mii in 3DS Community in AR

正因为 3DS 在 AR 社交上做出的种种突破性创新，使市场中的潜在掌机购买者纷纷置入新掌机。3DS 掌机让任天堂在当年面对拥有三国无双、讨鬼传等巨量 IP 的 SONY 旗下的 PS Vita 时大获全胜，销量遥遥领先。不仅如此，截止 2017 年，3DS 的生命力依旧没有枯竭，在销量上超过了传奇主机 PS2，跃升为游戏机历史第三销量，仅次于 GB 和 NDS。

表1　各大游戏机销量情况对比

Table1　Comparison of sales of major game consoles

截至 2017 年销量情况		
主机（发售年）	年销量	总销量
3DS(2011)	1,874,457	21,911,413
PSV(2011)	865,002	5,247,419
NDS		32,990,000
GB		32,470,000
PS2		21,833,261

由上述分析可知，将 AR 增强现实系统引入校园社区的建设中，可以有效地增强同学们使用此社区系统的兴趣，提高社区系统的使用率。

3　AR 交互设计

3.1　识别图与 3D 模型获取

3.1.1　思路介绍

首先，本交互系统意在使用每个人的校园卡当做个人角色进入界面的识别图。由于现阶段我校校园卡背面为同一图案，并不能做到每个人拥有其独特的识别图。

在之后的设想中，本系统建议将每人的校园卡背面放置一个能够明显识别的图案，比如个人信息的二维码等。通过扫描二维码，可以便捷地进入 AR 界面并识别出用户。

在现阶段调试以及模型初步构建中，本文使用简单的校园卡背面图案当做识别图、Unity3D 素材库中下载的 3D 模型当做学生人物建模角色。其后系统建设中，只要将识别图替换，并将模型替换为每个用户自己设置的模型即可。

3.1.2　识别图创建

本系统使用 Vuforia 自带的识别图创建功能制作识别图。

由于最新版校园卡的背面图案颜色色调单一，没有明显的识别点，故选择旧版校园卡背面图案制作识别图。

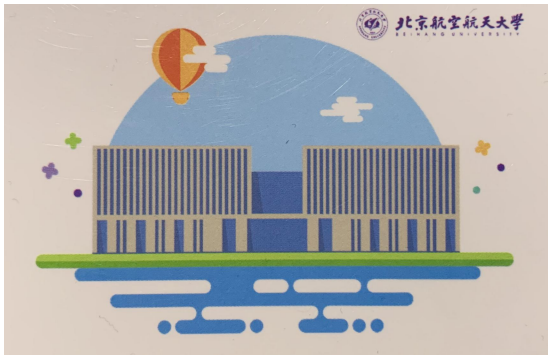

图 4　AR 模型识别图

Fig4　AR Model Recognition Target

本识别图通过手机摄像头拍摄，后期处理包括透视校正、增强对比度、鲜艳度以及清晰度，

使系统能够更快速地提取识别点。

3.1.3 3D 模型下载

本系统选择 Unity 官方 3D 素材库中经典的 3D 人物角色模型 Unity-Chan，本模型建模精细，功能齐全，素材库整齐。

当后期用户创建自己的个人角色模型时，只需改变原模型的素材库即可。

3.2 基础 AR 功能制作

首先制作基础的 AR 增强现实功能，即手机扫描识别图，识别图上出现 3D 模型。

由于本次编程使用的是 Unity2018.3.0f2 版本，以此版本经内置 Vuforia 的 SDK。所以只需导入识别图与 3D 模型两个 unitypackage 文件即可。

在主界面中删除原有 Camera，创建新的 ARCamera 与 ImageTarget，将 3D 模型放置在 ImageTarget 的目录下。随后将证书 Key 复制进根目录下，最基础的 AR 增强现实模型便制作完成。经检验，可以完成功能。

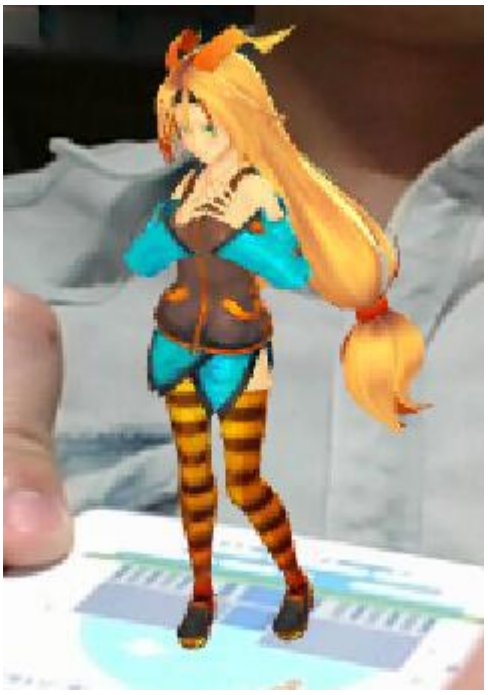

图 5　基本 AR 模型的实现

Fig5　Implementation of Basic AR Model

3.3 为模型添加目光追踪功能

从图 5 所示模型实现可以看出，若是仅仅能够实现模型的召唤，对于用户而言是没有丝毫体验感受的。

如果想要让用户感受到与 AR 模型的交互，需要 AR 模型对用户的动作做出最基本的反应。首先，便想到添加一个能够使模型的目光追随摄像头的功能。通过这个功能，可以使用户感觉时候收到 AR 模型的注视，拥有了最基本的交互感。

为了实现这个功能，本系统的思路如下：创建一个骨骼动画，使模型头部视角跟踪手机摄像头即可。

实现步骤如下：在模型界面创建新的 Layers，命名为 Head；在 Asset-FaceAnimation 文件夹下新建一个 Avatar Mask 遮罩，命名为 Head only mask。将此遮罩只适用于头部，并添加至 Head 层。在根目录下创建一个新的 C#脚本，通过脚本实现模型头部面朝方向锁定摄像头的功能。为了寻找合适的函数，通过 Unity 圣典查询到通过 IK 回调函数的定义以及调用方法，得知可以通过 IK 回调函数实现本功能。

在脚本编写完成后，首先将模型自带的两个动作脚本删除，再将新的脚本绑定至模型动作。经调试检验，可以完成功能。

图 6　AR 模型注视功能的实现

Fig6　Realization of gaze function in AR model

以下为 C#脚本代码：

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class unitylhtIK : MonoBehaviour
{
    public Camera cam;
    private Animator ani;
    void Start()
    {
        ani = this.GetComponent<Animator>();
    }
    void Update()
    {
    }
```

```
    void OnAnimatorIK()
    {
        ani.SetLookAtWeight(0.7f, 0.3f, 1, 1);
ani.SetLookAtPosition(cam.transform.position);
    }
}
```

3.4 为模型添加点击移动功能

3.4.1 点击移动

仅仅增加目光追随功能，对于线下社区交互系统而言，是远远不够的。为了实现最基本的交互系统，还至少需要添加模型的移动系统。

本系统选择了点击地面使模型移动的方式来实现模型移动。

为了实现这个功能，本系统的思路如下：从手机摄像头发射一束射线，射线与模型所在平面相交于一个坐标点，求出这个坐标点的坐标，并让模型移动到坐标处。

在制作过程中，将实现功能的过程分为三个步骤，首先简单的实现模型的位移，其次实现模型面向目标点位移，最后添加跑步动作进入位移过程。

首先，在实现位移过程时，关闭AR组件，在原有模型上新建平面，只做模型、地面、新建摄像机层面上的位移调试。

在分目录下新建一个C#脚本，并将其绑定至相机的脚本。用这个脚本实现移动功能。

以下为C#脚本代码：

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class MoveControllerLHT : MonoBehaviour
{
    public Camera cam;
    private Vector3 hitPoint;
    public GameObject girl;
    public float speed = 30;
    void Start()
    {
        cam = this.GetComponent<Camera>();
    }
    void Update()
    {
        if (Input.GetMouseButtonDown(0))
        {
            Ray ray =
cam.ScreenPointToRay(Input.mousePosition);
            RaycastHit hit;
            if(Physics.Raycast(ray,out hit))
            {
                if (hit.collider.gameObject.name ==
"Plane")
                {
                    hitPoint = hit.point;
                    Debug.Log(hitPoint);
                }
            }
        }
        girl.transform.position =
Vector3.MoveTowards(girl.transform.position, hitPoint,
Time.deltaTime * speed);
    }
}
```

经调试检验，此脚本可以满足功能：用鼠标点击Game界面内平面部分，模型移动至相应位置。

3.4.2 旋转移动

实现旋转移动思路：获取点击点位置坐标（之前函数已经实现），计算出目标点与当前点之间直线的角度值，将模型旋转至此角度。

为实现这个功能，只需将之前函数中的Update子函数稍作修改，修改代码如下：

```
void Update()
    {
        if (Input.GetMouseButtonDown(0))
        {
            Ray ray =
cam.ScreenPointToRay(Input.mousePosition);
            RaycastHit hit;
            if(Physics.Raycast(ray,out hit))
            {
                if (hit.collider.gameObject.name ==
"Plane")
                {
                    hitPoint = new
Vector3(hit.point.x, girl.transform.position.y, hit.point.z);
                    Debug.Log(hitPoint);
                }
            }
        }
        girl.transform.position =
Vector3.MoveTowards(girl.transform.position, hitPoint,
```

```
Time.deltaTime * speed);
            girl.transform.rotation =
Quaternion.Lerp(girl.transform.rotation,
Quaternion.LookRotation(hitPoint -
girl.transform.position), Time.deltaTime * 10f);
    }
```

经检验，此段函数能够大体实现功能。但存在一个小问题：模型到达终点后，无法自己固定角度，会自己再转回来，面向摄像头。

经分析，造成这个问题的原因是：获取角度时，若模型已经走到目标点，则目标点与现位置点连线向量为零向量。无法计算出角度，则角度归零，模型重新面向摄像机。

解决方法：设置一个角度变量，角度变量设置后不会即时清零。

设置角度向量：`private Vector3 look_rotation;`

更改函数：

```
if (hit.collider.gameObject.name ==
"Plane")
                {
                    hitPoint = new
Vector3(hit.point.x, girl.transform.position.y, hit.point.z);
                    look_rotation = hitPoint -
girl.transform.position;
                    Debug.Log(hitPoint);
                }
```

更改移动语句：

```
girl.transform.rotation =
Quaternion.Lerp(girl.transform.rotation,
Quaternion.LookRotation(look_rotation), Time.deltaTime
* 10f);
```

经调试检验，此脚本可以满足功能：用鼠标点击Game界面内平面部分，模型旋转至目标角度，移动至相应位置，移动结束后固定在当前角度。

3.4.3 添加动画

为实现此功能，需要新建animator controller文件，并将动作控制设置为如下图所示。

在WAIT与WALK两个动作的转移条件中，创建一个布尔值的变量，命名为“IsWalk”，当此变量为真时，动作由WAIT转换为WALK；当此变量为假时，动作由WALK转换为WAIT。

重新编辑C#动作脚本，以实现上述功能。

经调试检验，此脚本可满足功能：模型不动时，循环播放WAIT动作，点击某处开始移动后，开始播放WALK动作并移动至目的地。

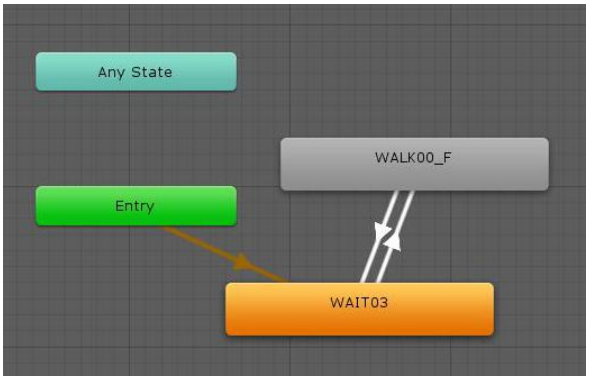

图 7　新建 animator controller 文件

Fig7　New animator controller file

最终版本C#脚本代码如下：

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class MoveControllerLHT : MonoBehaviour
{
    public Camera cam;
    private Vector3 hitPoint;
    private Vector3 look_rotation;
    public GameObject girl;
    private Animator ani;
    public float speed = 30;
    void Start()
    {
        cam = this.GetComponent<Camera>();
        ani = girl.GetComponent<Animator>();
    }
    void Update()
    {
        if (Input.GetMouseButtonDown(0))
        {
            Ray ray =
cam.ScreenPointToRay(Input.mousePosition);
            RaycastHit hit;
            if(Physics.Raycast(ray,out hit))
            {
                if (hit.collider.gameObject.name ==
"Plane")
                {
                    hitPoint = new
Vector3(hit.point.x, girl.transform.position.y, hit.point.z);
                    look_rotation = hitPoint -
girl.transform.position;
```

```
                    Debug.Log(hitPoint);
                }
            }
        }
        girl.transform.position =
Vector3.MoveTowards(girl.transform.position, hitPoint,
Time.deltaTime * speed);
        girl.transform.rotation =
Quaternion.Lerp(girl.transform.rotation,
Quaternion.LookRotation(look_rotation), Time.deltaTime
* 10f);
        if (Vector3.SqrMagnitude(hitPoint -
girl.transform.position) < 0.01f)
        {
            ani.SetBool("IsWalk", false);
        }
        else
        {
            ani.SetBool("IsWalk", true);
        }
    }
}
```

随后，将摄像机删除，重新启用AR增强现实，将模型与摄像机重新绑定，即可在AR界面实现模型的移动功能。

由于移动功能不能直观地用图片表示，便不再本文中展示。用户可以参考脚本实现。

3.4 为模型添加阴影

实现模型移动功能后，在使用过程中，发现模型对于现实场景的融合性还是不够强，不能给予用户足够的真实感。因此，尝试为模型添加阴影功能，增加模型的真实感。经查阅资料，确定实现阴影功能的步骤：

首先，新建一个摄像机，调整至假想光源的角度固定。将摄像机调整为 Soildbox 模式，调整为有一定透明度的黑色，新建Render Texture文件，绑定至摄像机。至此，Render Texture文件所保存的便是摄像机所拍摄的图片。

因为阴影只为角色创建，不希望角色脚下的识别图等其他物品也拥有阴影。因此为角色新建一个Layer，只让摄像机拍摄角色，将此Layer绑定至摄像机。

新建一个Plane，命名为shadow，给Plane新添材质球，设定为影子材质，将Render Texture贴到plane上。调整位置以及，使其初始位置与角色对齐。至此，便已经为角色添加了一个简略的阴影。

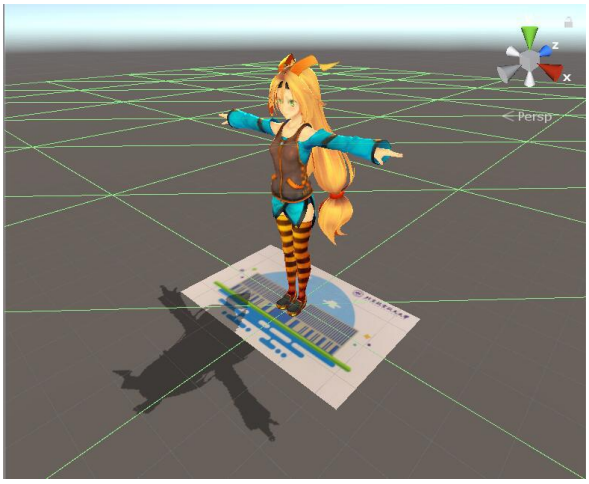

图 8 为模型添加阴影

Fig8 Adding shadows to the model

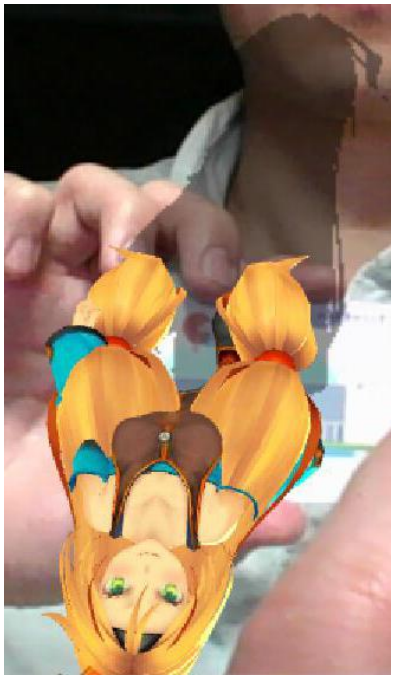

图 9 模型阴影的实现

Fig9 Implementation of Model Shadow

经调试检验，模型的阴影添加成功。阴影能够随模型运动而跟随运动。

至此，3D模型的设置以及脚本编写已经大体完成。

4 结 论

1) 通过 AR 增强现实，可以提高校园网络社区对大学生的吸引力，从而提高校园社区的参与率。

2) 本模型通过初步建立AR增强现实模型的交互。如目光跟随、点击平面移动、添加阴影，

能够很好地为用户提供沉浸感与交互感，进一步地吸引用户。

参考文献（References）

[1] 宋会玲，冀绚珂，孙慧娟.论大学生校园文化活动的参与度、口碑及期待值［J］.教育视点,2018,10:P15-P17.

[2] 时钟平.以 SNS 社区构建高校校园网络文化推进机制［J］.教学研究,2013,7:P9-P11.

[3] 卢曦.增强现实（AR）技术在大学校园的应用前景［J］.多媒体图形,2018:P126.

[4] 徐梦笛.论校园是增强现实发展的最好摇篮［J］.艺术科技,2017:P297.

[5] 顾长海.增强现实（AR）技术应用与发展趋势［J］.中国安防,2018,8:P81-P85.