每周工作总结

本周主要进行了以下工作：

1、 阅读 Unity 的官方文档，进行学习 Unity 的开发技术。

2、 进一步学习面向 Unity 平台的 C#代码开发。

3、 阅读增强现实应用相关的论文。

具体内容：

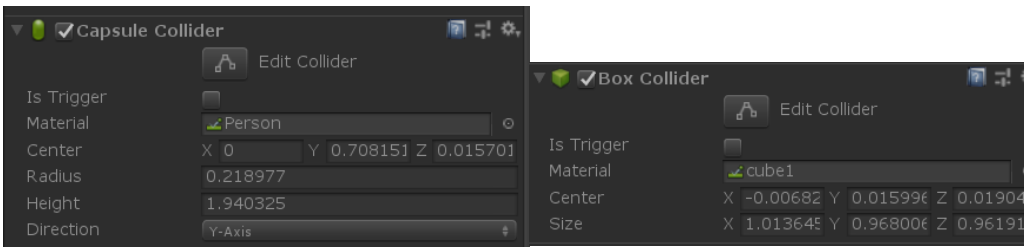
Unity 程序是通过模型+代码脚本的方式来运行的。通过在 Unity 中集成 Visual Studio，可以自动创建和维护 Visual Studio 项目文件。此外，在双击脚本或 Unity 控制台中的错误消息时，VisualStudio 将会打开。Visual Studio 的 C#编译器比 Unity 的 C#编译器目前支持的功能更多。也就是说，某些代码不会在 Visual Studio 中抛出错误，但在 Unity 中则会。Unity 会自动创建和维护 Visual Studio.sln 和.csproj 文件。每当在 Unity 中添加/重命名/移动/删除文件时，Unity 都会重新生成.sln 和.csproj 文件。也可以从 Visual Studio 向解决方案添加文件。Unity 随后会导入这些新文件，下次 Unity 再次创建项目文件时，便会使用包含的新文件进行创建。

模仿实现 Unity 中相机追踪目标的 LookAt 方法：

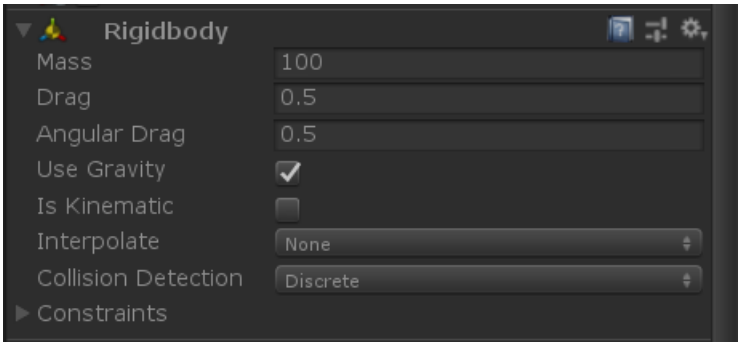
```
public class lookAtTest : MonoBehaviour
{
    public Transform target;
    void Start()
    {
        target.transform.position = Vector3.zero;
        transform.position = new Vector3(0, 30, 60);
        Vector3 diff = target.position - transform.position;
        Quaternionq Quaternion.FromToRotation(Vector3.forward, diff);
        Vector3 n = q * Vector3.forward;
        Vector3 worldUp = Vector3.up;
        float dirDot = Vector3.Dot(n, worldUp);
        Vector3 vProj = worldUp - n * dirDot;
        vProj.Normalize();
        float dotproj = Vector3.Dot(vProj, newUp);
        float theta = Mathf.Acos(dotproj) * Mathf.Rad2Deg;
        Quaternion qNew = Quaternion.AngleAxis(theta, n);
        Quaternion qall = qNew * q;
        transform.rotation = qall;
    }
}
```

利用 Unity 的 API 实现移动模型 A 碰撞到模型 B 的 5 秒后模型消失的功能。

1、分别设置模型 A 和模型 B 的碰撞体模型，并调整碰撞体模型的位置和大小



2、设置两个模型的刚体，可以在刚体的设置中调整模型的质量、阻力、角转动的阻力和是否使用重力等力学相关设置。



3、编写碰撞体之间有碰撞之后的操作的代码脚本，并将编写好的脚本挂在对应的模型上。

```
private void OnCollisionEnter(Collision collision)
    {
        if (collision.gameObject.name.Equals("Terrain"))
            return;
        else
        {
            Destroy(collision.gameObject, 5f);
        }
    }
```

还学习了其它的基础功能的开发操作，在此不一一列举。

阅读了题目为《基于沉浸式增强现实的流场可视化方法》的期刊论文，虚拟 物体在物理空间中的准确定位是增强现实领域中的关键问题，传统的增强现实定位技 术（如Vuforia）一般采用图像识别技术进行定位，通过物理空间中的特征图像定位虚拟物体位置。此类技术受到光照、特征图像污损等问题的影响，无法对虚拟物体进行精确的定位，这会对流 线在真实空间中的准确性造成影响。故针对虚拟物体无法精确定位，导致流线在真实空间中无 法准确反应流场运动规律的问题，提出自动+手 动的流线定位方法，以保证流线在现实空间中的准确显示。为减少用户手动定位工作量，系统在手动定 位前首先进行自动定位。自动定位采用传统的增 强现实定位方法，在物理空间中将特征图像放到 流场中心物体附近并记录相对位置（图中的特征 图像由 Adobe Illustrator 据 Vuforia 提供的模板 设计而成）。系统通过定位模块调用 头戴式增强现实设备相机在现实空间中检测特征 图像。在检测到特征图像后，根据特征图像与流 场中心物体的相对位置，即可计算出虚拟物体的 显示位置。

利用自动+手动的 定位方法，改善传统自动定位方法定位不准确，导致可视化结果不准确的问题。利用基于 Unity Mesh 的流管绘制算法在游戏引擎中完成流线绘制。最后基于该

方法实现了原型系统，为加快用户分析效率、提升用户体验，在该系统的基础上设计了基于用户凝视的流线布置方法与基于凝视与手 势的种子点放置方法。利用一组由飞机模型仿真计算生成的流场数据对系统进行了验证，并对该系 统的两种交互方式进行了用户评估实验。实验结 果表明系统能够利用上述方法在物理空间中准确 地完成流线绘制，反映流场变化情况。新的交互方 法使用户能够通过在真实空间的移动从不同的视角观察流场的变化情况，并使用凝视与手势等自然 交互方式快速、高效地改变流线在现实空间中的布 置，有效地提高了用户的分析效率，改善了用户的使用体验。