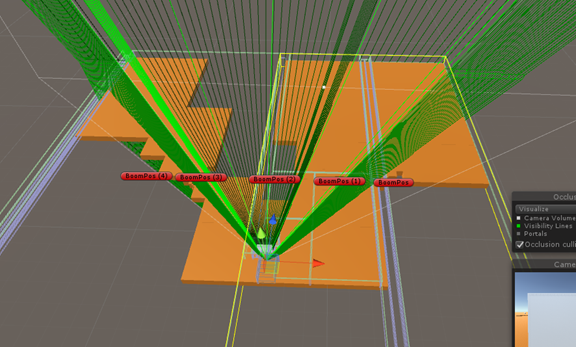
# **Unity完全自制游戏纸箱战争项目记录（20180630）**

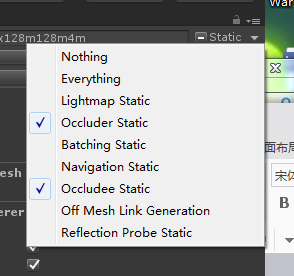
作者：[Pluto](http://gad.qq.com/user/index?id=2267380)

链接：<http://gad.qq.com/article/detail/48274>

今天是周六，依然进行了部分的制作，昨天采用了更为优化的爆炸方块替换效果，但对性能还是有不小的影响，昨天在临走之前听老师说可以使用遮挡剔除尝试一下，今天就研究了研究，经过查阅资料，实现了遮挡剔除的功能。

具体实现步骤是把需要进行遮挡剔除的方块在静态选项中勾选上Occluder Static和Occludee Static，在遮挡剔除面板中进行烘焙即可，使用起来很方便，在使用后就产生了如下的效果。





绿色区域为摄像机的可视区域，在可视区域以外的方块都不进行渲染，有效的提高了游戏运行效率。

新的问题随之产生了，因为游戏中需要切换载具，我发现在切换载具的时候，因为摄像机的变动，遮挡剔除似乎失效了，遮挡剔除似乎只能作用在一个摄像机上，这点问题存疑，还没有得到有效的解决办法。

因为地块崩开之后需要在一定的时间内删除，在Unity中即使物体被删除了仍然会在内存中占有一定的资源，想到利用对象池来解决这个问题，经查阅资料后，发现对象池并不适用与我设想中的地形崩坏，具体还得再次的考量考量

另外还写了一个通用的炸弹脚本，在游戏中所有的炸弹都可以用到，避免了再每个炸弹上单独挂载脚本的繁琐性。

using UnityEngine;

using System.Collections;

public class BoomScirpt : MonoBehaviour

｛

    //炸弹引爆时间

    public float BoomTime = 0;

    //炸弹爆炸产生伤害

    public int BoomHitHP = 200;

    //炸弹爆炸力量

    public float BoomPower = 2000.0f;

    //炸弹爆炸半径

    public float BoomRad = 5.0f;

    //炸弹爆炸掀起高度

    public float BoomUpPower = 500.0f;

    //炸弹是否允许多次爆炸

    public bool BoomRefresh = true;

    //炸弹是否可以对人体造成伤害

    public bool BoomDestructionHuman = true;

    //炸弹是否可以对载具造成伤害

    public bool BoomDestructionCarrier = true;

    //炸弹是否可以对环境造成伤害

    public bool BoomDestructionEnvironmental = true;

    //炸弹是否可以对地形造成伤害

    public bool BoomDestructionTerrain = true;

    public float BoomDestructionTerrainTime = 10.0f;

         void Start ()

    ｛

        StartCoroutine("BoomStartTime");

         ｝

         void Update ()

    ｛

        //测试功能

             if (Input.GetKeyDown(KeyCode.Q))

             ｛

            StartCoroutine("BoomStartTime");

             ｝

         ｝

    IEnumerator BoomStartTime()

    ｛

        yield return new WaitForSeconds(BoomTime);

        Vector3 BoomPos = transform.position;

        Collider[] HitCollider = Physics.OverlapSphere(BoomPos, BoomRad);

        foreach (var HitGameObject in HitCollider)

        ｛

            if (BoomDestructionHuman)

            ｛

            ｝

            if (BoomDestructionCarrier)

            ｛

            ｝

            if (BoomDestructionEnvironmental)

            ｛

            ｝

            if (BoomDestructionTerrain)

            ｛

                if (HitGameObject.gameObject.tag=="BigTerrain")

                ｛

                    HitGameObject.GetComponent<TerrainHP>().HP = HitGameObject.GetComponent<TerrainHP>().HP - BoomHitHP;

                    Debug.Log("boom");

                ｝

                if (HitGameObject.gameObject.tag == "SmallTerrain")

                ｛

                    if (HitGameObject.GetComponent<Rigidbody>()==null)

                    ｛

                        HitGameObject.gameObject.AddComponent<Rigidbody>().AddExplosionForce(BoomPower,BoomPos,BoomRad,BoomUpPower);

                    ｝

                    HitGameObject.gameObject.GetComponent<BoxCollider>().size=new Vector3(0.5f,0.5f,0.5f);

                    Destroy(HitGameObject.gameObject, BoomDestructionTerrainTime);

                ｝

            ｝

        ｝

        if (!BoomRefresh)

        ｛

            Destroy(gameObject);

        ｝

    ｝

｝