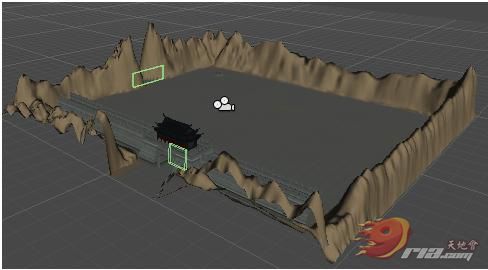
1、人物资源，需要有CharactorController，不然无法做object的Trigger碰撞响应。  
2、两个Cube，在其Inspector面板中，将BoxCollider下的IsTrigger属性勾选上，调整该Cube的形状，作为加载地形的触发点，我把它们放在地形交界点处，玩家碰触后加载对面的地形。  
   
3、将脚本绑定在两个cube上。  
4、把各个地形块都制作好后，将地形上的Object作为地形对象的子节点。在Project视图里，创建几个prefab文件，将地形文件分别拖入其中，然后制作成AssetBundle。  
5、制作过程中发现一个重大的问题，客户端模式Build出来的文件或者在编辑器中直观看到的结果，和Build成Web格式后的结果有些不一样。可能是WebPlayer的数据要依据固定的stream来源---WWW对象。而在CS脚本里所定义的成员或静态成员Object对象均无法保存当前WWW对象刚下载后的资源，造成下载部分和加载部分代码不能分开写，否则会因为丢失对象而加载失败，更糟糕的是在Web模式下，是不会报告什么错误的。所以这里解释下，两个脚本分别对应两个Cube的原因，就是各自保存了一份WWW对象，分别记录着自己所下载过的资源。注意不要用AssetBundleRequest对象来保存WWW对象中的asset，虽然手册上例子是这样写，但我们的问题也出现在这里，在经验和资料还不充分的情况下先做备案。

public class Script\_Cube\_Trigger\_Terrain1 : *MonoBehaviour* {

int Flag = 0; //记录是否已经加载过该资源，避免多次加载

//AssetBundleRequest abr; string debugstr = ""; //调试信息

static *WWW* wwwObj; //每个场景中一个单独的对象

void Start() {

if (Flag == 0) {

//判断平台信息，如果是webPlayer就从网络上加载，如果是本地，就从本地资源目录中加载。

if (*Application*.*platform* == *RuntimePlatform*.*OSXWebPlayer* || *Application*.*platform* == *RuntimePlatform*.*WindowsWebPlayer*) {

//实例WWW对象，相当于从参数代表的网络地址上下载资源过程。

//GlobalConfig.GetConnectIP()方法可以取得当前机器的全局IP地址（这个例子中，服务器端和客户端在同一台机器上）

wwwObj = new *WWW*("http://" + *GlobalConfig*.GetConnectIP() + "/AB/Terrain/Terrain1.unity3d");

} else {

// 因为玩家是玩Web形式发布的游戏，所以本地没有资源，所以这里可以省略了，本地调试的时候可以打开用。

// wwwObj = new WWW("file:///E:/123/Terrain1.unity3d");

}

}

}

// 当前对象（此例中就是那俩Cube）如果Collier的IsTrigger属性被勾选上了，那么当另一个Collier碰撞到此对象上，就会触发这个函数。

void OnTriggerEnter(*Collider* other) {

// this if-type hierarchy is just for simple debug, we can put debug message in each curly braces

if (Flag == 0) {

if (wwwObj != null) {

*print*(wwwObj.*isDone*.ToString()); //判断是否WWW对象已经下载完了

if (wwwObj.*assetBundle* != null) {

if (wwwObj.*assetBundle*.*mainAsset* != null) {

//load resouce into the assetbundle of wwwObject

wwwObj.*assetBundle*.*LoadAsync*("Terrain1", typeof(*GameObject*));

//创建地形实例

if (*Instantiate*(wwwObj.*assetBundle*.*mainAsset*) != null)

Flag = 1;

}

}

}

}

}

// 调试用的，通过GUI来调试，可视化，蛮方便的

void OnGUI() {

*GUI*.*Label*(new *Rect*(0, 0, 200, 200), debugstr);

}

}

后期期望：  
能实现本地缓存办法，让玩家下次登陆游戏时不用再去下载上次已经下载过的资源。  
能实现真正的后台下载，玩家登陆后再进行异步下载，在不影响玩家正常游戏的情况下按照指定的方案去下载。